SÓLO PROGRAMADORES

Revista especializada para usuarios de PC

AÑO 2. Nº 10
1250 PTAS.
ARGENTINA 9'50 $

Escrita por desarrolladores para desarrolladores

DE REGALO
CD-ROM con la versión educativa de WindowBase
+
Versión 2.20 del LINUX

Red Digital de Servicios Integrados

Terminal LINUX

Visual FOXPRO 3.0

Multitarea en modo protegido

Y además:

- Fichas de programación
- Tratamiento digital de la imagen
- Curso de programación
- Formatos gráficos
- Librerías FIVEWIN
- Curso de C++

9 778411 344792 00010

TOWER COMMUNICATIONS, S.R.L

SÓLO PROGRAMADORES
Revista práctica para usuarios de PC

JUNIO 1995. Número 10
Es una publicación de

TOWER COMMUNICATIONS, S.R.L.

Editor
Antonio M. Ferrer Abelló
Directora Comercial
Carmina Ferrer
Director de Producción
Carlos Peropadre
Publicidad
Magdalena Pedreño Llorente

••••••

Director
Mario de Luis García
Redactor Jefe
Carlos Doral Pérez
Coordinador técnico
Emilio Postigo Riancho

Colaboradores
Juan Manuel Martín, Luis Martín, Agustín Guillén, Carlos Arias, Enrique Coiras, Tomás Tejón, Sergio Rios, Fernando J. Echevarrieta, Francisco González, David Poves, Jaime Torroja, Carlos Ravina, David Aparicio, Fernando de la Villa, Francisco Monteagudo, Miguel A. Cazorla, Enrique Velasco, Fernando Pérez, Ignacio Cea.

Edición técnica
Emilio Castellano
Maquetación
Fernando García Santamaría
Tratamiento de imagen
Josefa Fernández Martínez
Servicios Informáticos
Digital Dreams Multimedia, S.L.
Ilustraciones
Miguel Alcón
Secretaría de Redacción
Consuelo Jiménez
Suscripciones
Erika de la Riva
Redacción, Publicidad y Administración
C/ Marqués de Portugalete, 10
28027 MADRID
Telf.: 741 26 62 / Fax: 320 60 72
Filmación
Duvial
Impresión
G.D.B.
Distribución
MIDESA

La revista SÓLO PROGRAMADORES no tiene porqué estar de acuerdo con las opiniones escritas por sus colaboradores en los artículos firmados. El editor prohibe expresamente la reproducción total o parcial de los contenidos de la revista sin su autorización escrita.

Depósito legal: M-26827-1994
ISSN: 1134-4792

SUMARIO

6 **NOTICIAS**
Las novedades más interesantes en el mundo informático para los programadores profesionales.

10 **WORLD WIDE WEB**
Una interfaz que permite moverse por Internet sin necesidad de conocimientos informáticos.

13 **FALLAS PARTY'95**
Reunión en Valencia de creadores españoles de DEMOS.

14 **ENTREVISTA**
John Bowne explica las ventajas de Window Base, el nuevo producto de su compañía.

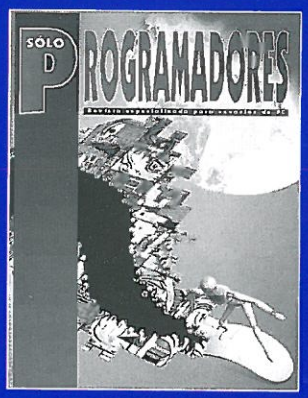
17 **CURSO DE ENSAMBLADOR**
Un acercamiento a la programación del coprocesador matemático.

24 **FORMATOS GRÁFICOS**
Estudio del formato BMP, fácil de manejar en el desarrollo de una aplicación Windows.

28 **CURSO DE PROGRAMACIÓN BÁSICA**
Antes de teclear un programa se debe realizar un nuevo análisis de los requisitos y problemas que pueden surgir durante la implementación.

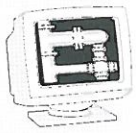
30 **HERRAMIENTAS GRÁFICAS**
En este artículo se verán técnicas para el realzado de imágenes, extracción de bordes y métodos de filtrado.

35 **COMUNICACIONES TELEFÓNICAS**
Una tecnología que funciona a 64000 bps parece destinada a reemplazar los modems: RDSI



SUMARIO

42 **LIBRERÍA FIVEWIN**
Gracias a esta librería es posible desarrollar auténticas aplicaciones Windows con CA-CLIPPER.



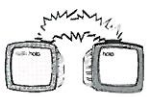
46 **MULTITAREA EN MODO PROTEGIDO**
El programa APDMX permite la ejecución simultánea de varias tareas dentro de un programa DOS.

48 **VISUAL FOXPRO 3.0**
Microsoft ha incorporado en este producto las características más solicitadas por los programadores: potencia y facilidad de manejo.

51 **SISTEMA OPERATIVO LINUX**
En este artículo se explica el modo de conectar unterminal a un PC ejecutando LINUX.

56 **PROGRAMACIÓN DE LA VGA**
El modo de vídeo 13h es el más sencillo de manejar de entre todos los que manejan 256 colores.

60 **REDES LOCALES**
En este capítulo se desarrolla una aplicación que permite el diálogo entre dos ordenadores mediante datagramas.



63 **PROGRAMACIÓN EN CLIPPER**
Este artículo estudia el uso de objetos para mejorar los mandatos de edición de campos y variables.

67 **SISTEMAS ABIERTOS**
Además de los filtros UNIX, se mostrará la utilización, formato y montaje de disquetes desde Linux.

73 **CURSO DE C++**
Consejos básicos a la hora de emplear este lenguaje.

78 **CONTENIDO DEL CD-ROM**
La última versión de Linux, junto con una versión reducida de WindowBase y los programas de comunicaciones TERMINATE y VTX.EXE.

81 **CORREO**
Respuestas a las preguntas planteadas por nuestros lectores.

VARIOS LIDERES DE LA INDUSTRIA ESTABLECEN UNA NORMA DE CONECTIVIDAD "PLUG AND PLAY" PARA LOS PC

IBM, Intel, Microsoft, Compaq, Digital, NEC y Northern Telecom han anunciado USB (Universal Serial Bus), una nueva norma de conectividad en bus serie. Con la llegada de esta norma se pretende mejorar las prestaciones y hacer más sencillas las conexiones de periféricos a PC y de red telefónica a PC.

USB permite transferencias a 12 megabits por segundo, presenta una arquitectura de concentrador en estrella que posibilita la conexión de un máximo de 127 periféricos a un sólo puerto PC, y autoconfiguración automática que reconoce y configura cualquier periférico externo conforme con USB. También proporciona soporte de nuevas capacidades telefónicas como la transmisión de sonido, imágenes y datos sin necesidad de conexiones telefónicas especializadas.

Para más información:
World Wide Web en http://WWW.teleport.com/~USB
Contacto de prensa:
Liliane Chinyavong
Tel: (91) 594 43 53

NUEVA TARJETA PEGASUS II PARA SALIDA DE VIDEO

El centro Informático de Boadilla S.L. y Electronic Design Ibérica S.L. anuncian la disponibilidad de Pegasus II entre su gama de productos, una nueva tarjeta para videoproducción profesional y semiprofesional informatizada.

Esta tarjeta de video ofrece una calidad de color de 24 bits, escalado de la imagen, filtro antiflicker, señal alphakey, método de doble buffer y posibilidades de genlocking. Existe un modelo interno para PC y pronto aparecerá en formato externo para ordenadores Macintosh y Silicon Graphics.

La tarjeta está disponible desde 119.900 pesetas y es posible añadir genlocks externos a partir de 69.900 pesetas.

Para más información:
Electronic Design Ibérica S.L. (Centro Informático Boadilla S.L.)
Convento, 6 1º 4ª
28660 Boadilla del Monte (Madrid)
Tel: (91) 632 23 75 / 27 65
Fax: (91) 632 10 99

NOVELL EDUCATION ALCANZA UN TOTAL DE 10.000 CNEs EN EUROPA

Novell Education continúa su crecimiento, alcanzando la cifra de 50.000 Ingenieros Certificados Novell (CNE) en todo el mundo. La zona de EMEA, formada por más de 50 países de Europa, Asia Central y Africa, ha llegado a un total de 10.000 CNEs.

Estos ingenieros certificados poseen probada experiencia en conceptos de soporte y redes, realizan tareas de instalación, soporte técnico y servicios de actualización.

Novel Education, cuyo objetivo es dirigir la Informática Universal global mediante programas de enseñanza de alta calidad, prevé un incremento en la tasa de crecimiento del 60 al 80 por ciento al año basándose en el ritmo de crecimiento actual.

Para más información:
Novell/WordPerfect Spain
Daniel Toledano
Tel: (91) 577 49 41

SYBASE CONSIGUE EL MAXIMO RENDIMIENTO TPC SOBRE EL ALPHASERVER 2100 DE DIGITAL

Sybase y Digital Equipment Corporation han anunciado la conse-

cución del mejor rendimiento TPC-C por Sybase SQL Server System 10. Este logro, auditado por el Transaction Processing Performance Council, ha sido alcanzado sobre el sistema AlphaServer 2100 de Digital.

La velocidad conseguida fue de 1708,12 transacciones por minuto (tpm-c) con una relación precio/rendimiento de 555 dólares por tpm-c.

El sistema AlphaServer 2100 4/275 utilizado dispone de 4 procesadores, con sistema operativo UNIX 3.2 de Digital, SQL Server 10 de Sybase y el sistema Tuxedo de procesamiento de transacciones UNIX de Novell

Contacto de prensa:
S.E.I.S.
Marcos Carmena
Tony Noel
Tel: (91) 308 43 15
Fax: (91) 410 38 76

LOS PRINCIPALES FABRICANTES APUESTAN POR SYBASE SQL SERVER MONITOR PARA DESARROLLAR SUS PRODUCTOS

Según Sybase, los principales fabricantes de gestión de sistemas que participan en su programa Systems Management Alliance, utilizan el API de Sybase SQL Server Monitor para el desarrollo de sus productos. SQL Server Monitor proporciona información detallada sobre el rendimiento de bases de datos Sybase.

El API ofrece un objetivo de desarrollo estable y facilita la creación y mantenimiento de soluciones para la gestión de sistemas, aparte de facilitar a los usuarios finales de Sybase la creación y el mantenimiento de sus propias aplicaciones.

Como ventaja adicional, dicho API no se verá afectado por las nuevas versiones del código de SQL Server.

Contacto de prensa:
S.E.I.S.
Tel: (91) 308 43 15
Fax: (91) 410 38 76

NUEVA VERSION DE LA HERRAMIENTA DE PROGRAMACION VISUAL VBTOOLS

DanySoft Internacional ha anunciado la disponibilidad de la herramienta de programación visual VBTools 4 para Visual Basic, Visual C++ y cualquier lenguaje que soporte el tipo 1 de VBX.

Esta herramienta de MicroHelp incluye 22 controles que pueden incorporarse al control de datos de Visual Basic, 54 controles de usuario, mejoras de los controles 3-D, y la tecnología del Grid/VBX de FarPoint incluida en cada copia.

Por otro lado, DanySoft también ha anunciado la herramienta TrueGrid Pro para Visual Basic, un producto que facilita la creación de browses de bases de datos totalmente configurables y editables.

Para más información:
DanySoft Internacional
Paseo de Albacete, 73
28700 San Sebastián de los Reyes (Madrid)
Tel: (91) 663 86 83
Fax: (91) 654 63 82

CISCO AUMENTA SU OFERTA RDSI

La compañía Cisco Systems anuncia la mejora de las capacidades RDSI de dos de sus plataformas, con la incorporación de interfaces RDSI PRI (acceso primario) con velocidad E1 a la familia de routers CISCO 7000, y con la incorporación de interfaces RDSI BRI (acceso básico) al router/hub Cisco 2516.

El soporte RDSI PRI E1 (2048 Mbps) ha sido añadido mediante una tarjeta de acceso WAN de alta densidad, el MultiChannel Interface Processor (MIP). PRI E1 es compatible con el router AccessPro con soporte RDSI y con la línea completa de productos RDSI de Cisco.

La nueva versión del router/hub integrado Ethernet dispone de un puerto RDSI BRI, dos puertos serie síncronos, un puerto serie asíncrono y 14 puertos Ethernet. A diferencia de otros productos, el modelo 2516 no sacrifica sus puertos serie síncronos para conseguir la funcionalidad RDSI.

Cisco Systems también ha anunciado la mejora del Sistema Operativo de Interconexión de Redes (IOS) de Cisco mediante la adición de nuevas capacidades específicas RDSI, como la señalización PRI E1 y la compresión de datos PPP.

Contacto de prensa:
S.E.I.S.
Mario Gutiérrez
José González Rivera
Tel: (91) 308 43 15
Fax: (91) 310 38 76

IBM OFRECE EL ACCESO A INTERNET EN ESPAÑA MAS BARATO Y CON MAYOR COBERTURA

IBM amplia la cobertura de acceso a la red Internet a través de la red internacional IBM Global Network. Dicha red cuenta con 17 nodos de conexión en toda España, ofreciendo a los usuarios un acceso más rápido y económico mediante llamadas telefónicas locales.

Se ofrecen dos posibilidades de acceso a los servicios. Una está basada en el sistema operativo OS/2 Warp, que incluye sin coste adicional el software necesario. La segunda posibilidad está dirigida a los usuarios de ordenadores con Windows. Este software se distribuirá de forma gratuita en toda España a través del teléfono del Canal Directo de IBM.

La oferta actual incluye inscripción gratuita al servicio durante 1995, con una cuota fija de 2200 pesetas al mes con tres horas de acceso gratuito, y un coste de 580 pesetas por cada hora adicional de conexión.

Por otra parte, IBM ha anunciado la incorporación de novedades a su oferta de acceso a Internet durante el presente año.

Para más información:
Canal Directo IBM
Tel: 900 100 400
E-mail: informa@es.ibm.com

BREVES

UNISYS, NUEVO CENTRO DE SOPORTE AUTORIZADO DE MICROSOFT

Unisys, como nuevo Centro de Soporte Autorizado Microsoft (ASC) proporcionará soporte a los usuarios corporativos de Microsoft en sistemas multivendor desktop y redes. El nombramiento de Unisys como ASC ha sido realizado en el contexto del programa Microsoft Solution Providers, que cubrirá 15 países europeos.

Para más información:
UNISYS
Flora Rué
Tel: (91) 721 12 12

ACUERDO DE COLABORACION EN SISTEMAS ABIERTOS ENTRE SSA Y CP SOFTWARE

Sistemas Industriales de CP Software y SSA Ibérica han llegado a un acuerdo para la comercialización del sistema de gestión de empresas cliente/servidor BPCS, disponible en sistemas UNIX y AS/400. El acuerdo se extiende a la comercialización de herramientas de ingeniería AS/SET y AS/Vision y productos de intercambio electrónico de datos (EDI).

Para más información:
Tel: (91) 413 86 96
Fax: (91) 413 81 99

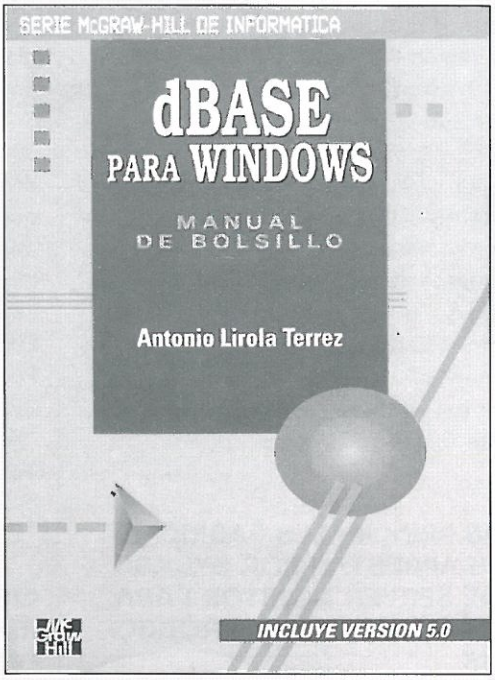
LA ASOCIACION APSE, A DISPOSICION DE LOS PROGRAMADORES ESPAÑOLES

La asociación APSE ofrece sus servicios para que la programación de shareware en España alcance la importancia que tiene en otros países. Para ser miembro de la asociación es preciso dejar la versión shareware de un programa propio del mismo tipo en CiberLinea BBS (902 23 86 24). Si el programa se considera válido, se ofrece pertenecer a APSE durante un año con acceso a todos los recursos de su BBS.

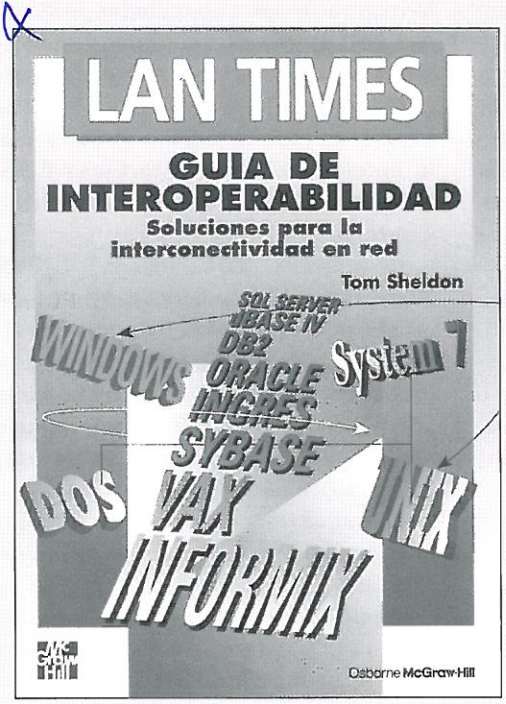
LIBROS

DBASE PARA WINDOWS. MANUAL DE BOLSILLO

A diferencia de otros manuales de bolsillo, este libro no pretende enseñar el manejo de un programa partiendo de cero. Se trata de una pequeña obra de consulta, dedicada a dar una breve explicación de las características y funciones de uso más habitual en dBase para Windows. El libro se compone de dos partes claramente diferenciadas: en la primera se describen los conceptos básicos necesarios, y en la segunda se exponen las características y funciones, ordenadas de forma alfabética para facilitar su consulta.



Editorial: McGraw-Hill (91) 372 84 10
Autor: Antonio Lirola Terrez
238 páginas
Precio: 3.700 pts IVA inc.



ENCICLOPEDIA LAN TIMES DE REDES (NETWORKING)

El título de Enciclopedia de Redes hace honor a esta obra, por su tamaño y contenido. El tratamiento de cada tema se hace en profundidad, acompañando al texto con multitud de esquemas y dibujos que ayudan a la comprensión de cada punto. En sus páginas se tocan temas tan dispares como el cableado, los protocolos, las tecnologías, la seguridad y las últimas tendencias en el complicado mundo de las redes informáticas. Los contenidos del libro se presentan ordenados alfabéticamente, y al final se incluye un glosario inglés-español.

Editorial: Osborne/McGraw-Hill
Autor: Tom Sheldon
1188 páginas
Precio: 4.300 IVA inc.

DE CLIENTES E INFORMÁTICOS

Yo lo sé. Yo sé lo que hay que hacer. Pero cada vez que intento vérmelas con un problema técnico, algún maldito imbécil quiere que yo tome una decisión sobre camiones, o teléfonos, o alguna otra condenada cosa.

Robert Heinlein - El hombre que vendió la Luna

Se preguntarán a qué viene esta cita. Pero primero tengo que hablarles de mi mujer. Mi mujer es lingüista. Yo soy informático. Esto es lo que aporta complementariedad a nuestro matrimonio, y no, como diría un ex-novio suyo que siempre me ha caído gordo, el hecho de que su signo zodiacal sea de fuego y el mío de agua. La nuestra es una pareja tan basada en la cuadratura del círculo que mi media naranja acudió a un congreso de Lingüística sólo para presenciar una ponencia titulada "¿Pueden los lingüistas y los informáticos entenderse?". El ponente no se presentó. Posiblemente tuvo que acudir a una reunión urgente con el abogado de su señora.

El otro día, ella volvió muy excitada de una entrevista de trabajo. Iba a participar como autoridad lingüística en la elaboración de un CD-ROM, y el lunes siguiente tendría lugar la reunión inicial entre todos los miembros del proyecto. Yo me apresuré a decir:

— Ten mucho cuidado con los informáticos.

— ¿Por qué?

— Te mirarán con suficiencia, te bombardearán con palabras técnicas para tomar posiciones, y a cualquier cosa mínimamente creativa que les sugieras, te responderán abriendo mucho los ojos, como si hubieras dicho una barbaridad, y dirán muy serios, meneando lateralmente la cabeza: "No, eso no se puede hacer".

— Pues son una compañía americana bastante potente, con mucha experiencia en estas cosas.

— Pues más a mi favor.

La entrevista tuvo lugar, y mi cotitular de Libro de Familia volvió dándome la razón. La verdad es que yo había sido cautelarmente pesimista. Sólo quería que estuviera preparada por si se encontraba con informáticos de la variedad recalcitrante. Pero así fue. Palabra por palabra. No le dijeron: "Mm, tendremos que pensarlo", ni "Para estas cosas tendríamos que reelaborar el presupuesto". Dijeron exactamente lo que previ: "No, eso no se puede hacer". Y yo sabía que las propuestas eran razonables. Menos mal que ella tenía el antídoto preparado. Les dijo: "Bueno, en la próxima reunión traedme fotocopias del manual de la herramienta de desarrollo, para que yo me vaya haciendo una idea de lo que se puede hacer y lo que no".

Sus posiciones se flexibilizaron de inmediato.

Hay que decir en defensa de los desarrolladores —y por tanto, en la mía propia— que esta actitud de esconderse en el caparazón es fruto del roce con clientes, que han convertido un proyecto que se tenía por sencillo en un infierno. Y lo han hecho a base de pretensiones fantasiosas, especificaciones difusas y, sobre todo, volantazos a última hora en mitad del proyecto. Todos los que hemos trabajado de esta manera, sabemos la cantidad de código que ha habido que tirar a la papelera por culpa de alegres cambios de concepto. Y en la mayoría de los casos, todo esto ocurre porque el cliente no sabe tratarnos, y nosotros no sabemos tratar con el cliente. En un esfuerzo de buena voluntad, y antes de que el cliente y nosotros nos echemos las manos al cuello, allá van un par de indicaciones que quizá sirvan para mejorar nuestra comprensión mutua.

LO QUE PIENSA EL INFORMÁTICO DEL CLIENTE

Este tipo (o tipa) no tiene ni idea de informática, y se le nota. No sabe expresar sus necesidades en términos no ambiguos. No tiene ni idea de lo que el ordenador puede hacer y lo que no. Alguien le ha dicho que "en informática todo es posible" y no tiene sentido alguno de la medida de lo que cuesta hacer las cosas de una forma u otra. Tendré que ir haciéndolo como Dios me dé a entender, y cuando lo tenga casi todo terminado, me dirá que le haga "pequeños cambios" que me obligarán a reescribirlo todo.

LO QUE PIENSA EL CLIENTE DEL INFORMÁTICO

Esta tipa (o tipo) no tiene ni idea de camiones (o de teléfonos, o de alguna otra condenada cosa), y se le nota. Sólo sabe pensar en sus incomprensibles términos de megabytes, entornos, herramientas y análisis. Yo le explico lo que quiero, y no quiere entenderme. Yo no quiero una base de datos, ni dos, ni tres. No quiero usar un intérprete ni un compilador. No quiero ordenar por el método de la burbuja ni por *Quicksort*. ¡Quiero que resuelva mi problema! Y cada vez que intento explicarlo, me responde con "tecnovaguedades".

LO QUE PUEDE OFRECER EL INFORMÁTICO AL CLIENTE

Mira, lo más importante en una realización informática es que tú y yo tengamos muy claro, y tan bien especificado como sea posible, qué es lo que queremos hacer. No te preocupes si ahora mismo no sabes decirme muy bien qué quieres: es normal. Ahora mismo, nos sentamos tú y yo en esta mesa y yo te ayudo a definirlo, y de paso intento adquirir algo de tu conocimiento sobre camiones (o sobre teléfonos, o sobre alguna otra condenada cosa). Y lo hacemos con mucho papel, muchos lápices y unos bocadillos. Nada de ordenadores. No pulsaremos ni una sola tecla hasta que hayamos superado esta etapa. Cada uno de los días que pasemos haciendo esto, nos estaremos ahorrando una semana de trabajo más tarde. Cuando estemos satisfechos, empezaremos la segunda —y más fácil— parte del trabajo. Entretanto, cada vez que yo suelte un tecnicismo de los míos, tú me das un tirón de orejas. Y cada vez que tú me hagas lo mismo, yo te pellizco la nariz.

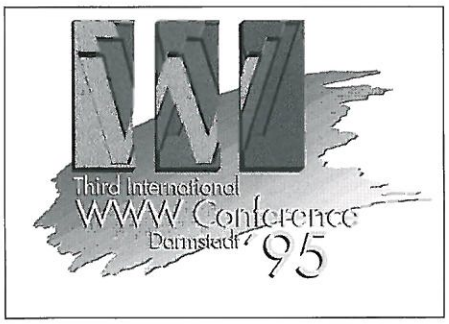
¿Comprenden ahora el porqué de la cita inicial? Pues hale, a entenderse. ¡Primer round... Doonng..! •

WORLD WIDE WEB

3ª CONFERENCIA INTERNACIONAL DE WORLD WIDE WEB

Fernando J. Echevarrieta

Todo el mundo ha oído hablar de Internet sin embargo, hasta ahora, sólo en unos pocos países como EE.UU. ha llegado hasta el ciudadano medio. Dejando a un lado razones económicas (Sres. de Telefónica, una llamada de atención), uno de los principales motivos es que el sistema debe ser algo tan sencillo que cualquiera pueda manejarlo. Ahora, con el WWW se ha resuelto este problema. Sin embargo, cada fabricante desarrolla sus propias soluciones, extiende a su modo los protocolos tratando de competir por los primeros puestos en el mercado. Por ello y, especialmente en este terreno que avanza a una velocidad vertiginosa, es necesario organizar una serie de reuniones periódicas para presentar los avances y discutir los procesos de normalización.



El logotipo de la Third International WWW Conference, Darmstadt '95, Alemania.

El WWW supone una interfaz que permite navegar virtualmente a cualquier persona por Internet sin necesidad de ningún tipo de conocimientos sobre informática. Por su facilidad de uso, cualquier hogar es ya un posible mercado para esta red Internet. Por ello ha comenzado una carrera contra reloj en la que nadie quiere quedar atrás, pero aún hay muchos problemas por resolver.

LA GRAN TELA DE ARAÑA (O EL WWW)

World Wide Web (WWW, W3 o W^3) o la "tela de araña de cobertura mundial", es hasta ahora el sistema de mayor éxito para el acceso y la distribución de información en Internet. Su éxito se sustenta en el apoyo que tiene en hipermedia. Así como en un sistema hipertexto (como los ficheros de ayuda de Windows). Algunas palabras del texto aparecen destacadas, indicando que si se hace click con el ratón sobre ellas conducirán a un nuevo documento, un sistema hipermedia es, por extensión, aquel que interconecta todo tipo de información multimedia, a través de información también multimedia. Así, por ejemplo, pulsando sobre la fotografía de una persona en un documento se puede acceder a información sobre ella, ya sea en audio, vídeo, texto u otro documento combinando todos ellos. Del mismo modo, cada ítem de información multimedia en el nuevo documento puede ser un enlace a otros, y cada información provenir de una máquina distinta de Internet, de un lugar diferente del mundo que ha quedado enredado en la tela de araña (WEB) que constituyen todos los enlaces hipermedia.

El WWW es el gran culpable del gran interés que por Internet se ha despertado recientemente, ya que a través del mismo se consigue que el usuario final no necesite poseer conocimientos sobre informática, ni sobre los protocolos de la gran red. Simplemente se necesita disponer de un programa cliente de WWW o browser, de los que ya existen numerosas versiones para todas las plataformas. De esta forma, cualquier persona podrá encontrar la información que desea navegando por este libro multimedia que supone el WWW con sólo hacer click con su ratón y, si lo que quiere es transferir un fichero, bastará con que pulse sobre su nombre. Es por su sencillez, por lo que, en ocasiones se confunden los conceptos de WWW e Internet y por lo que el Web ha desplazado a todos los demás sistemas de búsqueda de información en "la red" ya que, además, es capaz de proporcionar pasarelas a lo mismos con sólo pulsar el botón del

La Universidad Técnica (TH), sede de la conferencia, fue fundada en 1836 y cuenta en estos momentos con unos 12.000 alumnos.

ratón. Por estas razones, se puede decir que el Web es la interfaz, la puerta de entrada a Internet para el usuario final y por ello, mueve grandes intereses, muchos de ellos económicos que llevan a organizar unas dos conferencias internacionales al año.

TERCERA CONFERENCIA INTERNACIONAL SOBRE WWW

Durante los pasados días 10 a 14 de Abril tuvo lugar la Third International WWW Conference en la Technical University Darmstadt (THD) en Darmstadt, Alemania. Con un completo programa de tutoriales, mesas de trabajo, sesiones técnicas, conferencias, mesas redondas, presentaciones de trabajos y exposiciones, la conferencia, organizada por el Fraunhofer Institute for Computer Graphics, resultó uno de los principales eventos acerca de aplicaciones de Internet sobre el World Wide Web. A ella acudieron alrededor de 3000 participantes de más de 40 países.

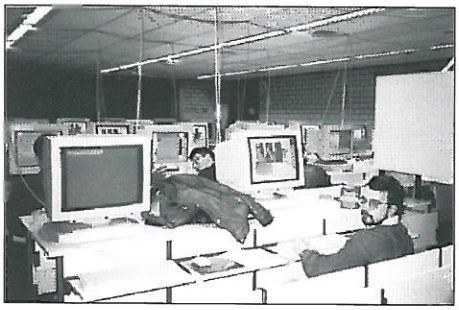
En los ciclos de conferencias y mesas redondas se trataron temas como la utilización comercial del WWW en Europa, destacando las distintas circunstancias respecto al mercado americano; sistemas de información en empresas de cobertura mundial; marketing sobre el Web; seguridad en el Web; estandarización e innovación; SGML y el Web, y propiedad intelectual y seguridad.

Especialmente interesante fue una mesa redonda sobre la próxima versión 3 de HTML (HyperText Markup Languaje) o HTML+, el lenguaje de hipertexto, y las funciones que debería incluir para la presentación de matemáticas en la que, junto a informáticos y matemáticos, el público asistente manifestó sus propuestas.

Tras la primera jornada de trabajo, tuvo lugar una recepción.

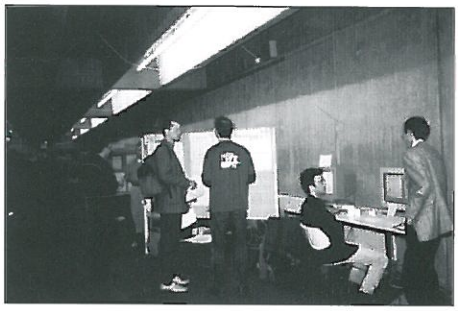
Aunque la conferencia que lo presentó dejó mucho que desear, es de destacar el mecanismo incorporado en las últimas versiones de browsers de NCSA Mosaic, el CCI (Client Communication Interface), que permite a otras aplicaciones comunicarse con el browser, permitiendo que éste actualice la información que muestra en tiempo real, o ejecute determinadas acciones sin necesidad de intervención humana, por poner dos ejemplos.

Durante los días en que transcurrió el evento, se puso a disposición de los asistentes dos salas con puntos de acceso a Internet, con la finalidad de proporcionarles experiencia de primera mano sobre los temas presentados, así como de permitirles acceder a sus respectivos sistemas para sentirse como en casa. Asimismo, toda la información de última hora, noticias, cambios de programa y otros eventos, se reflejaban instantáneamente en el servidor de la conferencia que podía ser consultado a través de estos puntos de acceso a Internet y de los de todo el mundo.



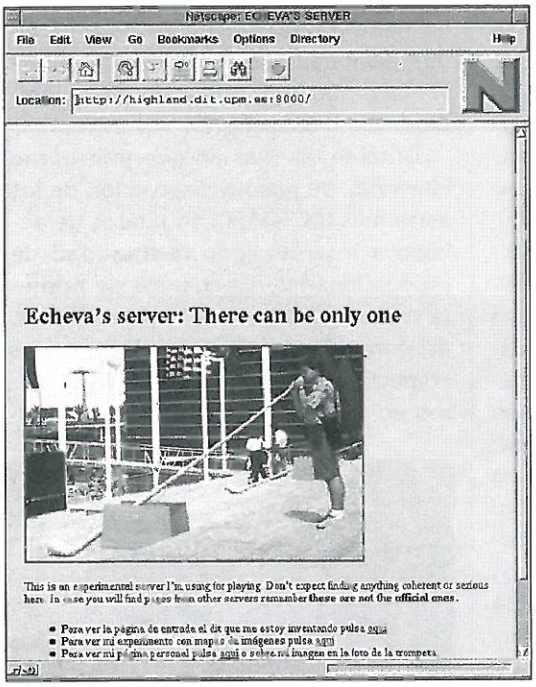
Durante las jornadas, los asistentes tuvieron a su disposición dos salas con estaciones de trabajo para acceder a Internet.

Es necesario hacer también mención de la presentación de Microsoft sobre su tan esperado sistema operativo Windows '95. Anecdótico el momento en que, al pasar a la demostra-



Paralelamente a los ciclos de conferencias y mesas redondas, se celebraba una exposición de distintos proyectos de universidades y empresas.

ción proyectada sobre pantalla gigante, en lugar del potente sistema aparecido un mensaje de error, lo que despertó los aplausos y carcajadas del público asistente. El error, ajeno a Microsoft, se debió al sistema de proyección y una vez subsanado, la demostración se llevó a cabo de forma correcta. En cualquier caso, el público mostró en reiteradas ocasiones su impaciencia ante los retrasos en la comercialización del sistema, interrumpiendo la presentación a destiempo, con sarcásticos aplausos.

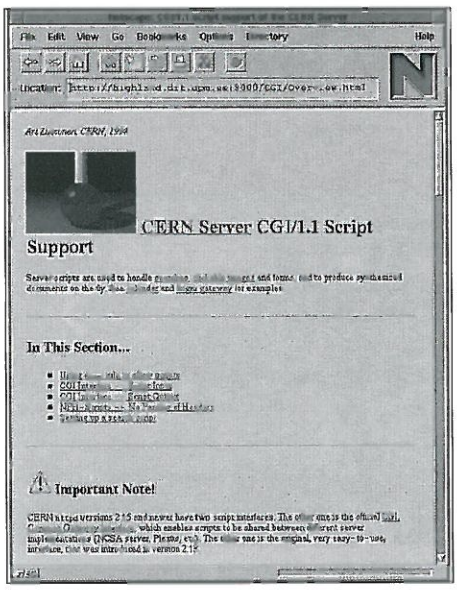


El WWW permite que con suma facilidad cualquier usuario pueda crear su propio servicio en Internet.

EXHIBICIONES Y POSTERS

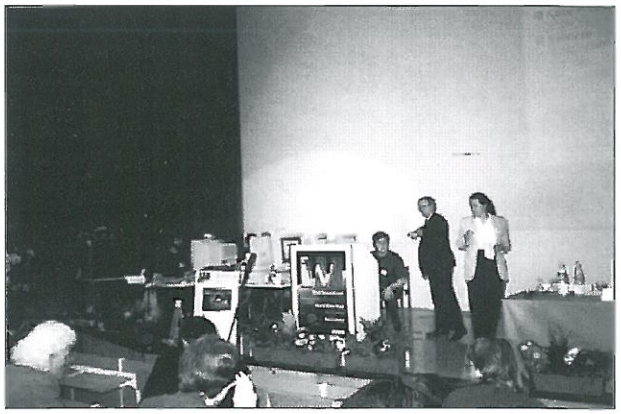
A la conferencia acudieron también veinte expositores con la finalidad de presentar sus últimos productos, así como los proyectos en desarrollo: Apple Computer, EMWAC, Exoterica Corporation, Fraunhofer Institute for Computer Graphics, Fulcrum European Headquarters, GRIF, INRIA, MID / Information Logistics Group, NETMANAGE Software, O'Reilly International, O2 Technology, OCLC, Process Software Corporation, Silicon Graphics, Softquad Inc., Spyglass Inc., Sun Microsystems, Telequip y Verity. Asimismo, en otra zona del recinto se realizaban presentaciones de otros sistemas y proyectos en desarrollo realizados por otras empresas, universidades e instituciones públicas.

En la exposición, llamó especialmente nuestra atención la presentación del browser HotJava de SUN. El browser, desarrollado en lenguaje Java de SUN (una evolución de C++), define una máquina virtual independiente del hardware, que es capaz de ejecutar programas que le envía el servidor WWW o que se le enlazan. De esta forma, es capaz de autoadaptarse a extensiones no estándar de los protocolos que definan distintos fabricantes así como de "aprender" a mostrar cualquier formato presente o futuro, de objeto hipermedia. Del mismo modo, dentro del mismo documento de Web, es posible observar la ejecución de programas interactivos. El producto despertó tal interés que se organizó un seminario no programado sobre el mismo el último día de conferencias.



A través del browser podemos hacer "surfing or navigating the Internet" (surf o navegar por Internet).

Otro producto a destacar fue WebFORCE de Silicon Graphics, que combina software servidor y de edición con las prestaciones de la estación de trabajo Indy. Para crear documentos de WWW, hasta la aparición de editores como éste, era necesario conocer HTML, por lo que herramientas de estas características, basadas en interfaces drag & drop, facilitan cada vez más la comunicación en Internet no sólo al usuario cliente sino también al administrador del sistema.



Momentos antes de una conferencia presentada por INRIA y SGI se realizan los ajustes necesarios en los ordenadores del Auditorium Maximum.

En la carrera por conseguir el mercado de los browsers, Mosaic avanzó un paso para recuperar el terreno que Netscape le había ganado, al presentar la nueva versión de Enhanced Mosaic de Spyglass Inc.

Y, EL FUTURO...

La "Conference" quedó abierta. El acto de clausura, más que un cierre, supuso una nueva cita para la Fourth International World Wide Web Conference, Boston'95, EE.UU., que se celebrará a finales de año. Para entonces se sabrá cuales de las propuestas realizadas en Alemania han llegado a cuajar, y es probable que también para entonces, la Internet comience a llegar a los hogares españoles, de la misma forma que el año pasado lo hizo el CD-ROM. Esperemos que así sea, y todos podamos asistir a través del WWW a la Fifth Conference del 96 conectados por módem desde nuestros hogares. ■

LA DEMOMANÍA CRECE EN ESPAÑA

Francisco González

David Poves

Estos programas son normalmente realizados por grupos en los que unen sus fuerzas programadores, grafistas y músicos. Un ejemplo pueden ser Future Crew o Iguana, que son uno de los grupos más conocidos internacionalmente.

A quien le gustan las demos conoce los "parties" o reuniones que se celebran especialmente en los países nórdicos como "Assembly" (Sólo Programadores nº 2), "The Party" o "The Gathering" en Finlandia, Dinamarca y Suecia respectivamente. En estos "parties" se reúnen los grupos que crean las demos para presentar sus nuevas producciones a concurso, conocerse entre ellos, o simplemente para pasar un rato divertido entre gente que comparte su afición. Durante los días 24, 25 y 26 del pasado mes de Marzo, se celebró en España una reunión de ese tipo, aunque a menor escala. Fue concretamente en la Extensión del I.B. San Vicente Ferrer de Valencia, cuya junta directiva cedió desinteresadamente algunas de sus aulas y concedió un presupuesto para el encuentro . El nombre de la reunión fue "Fallas Party '95", organizada por el grupo español "The Lords Of The Bits" y contando con la colaboración de Tower Communications, que proporcionó regalos a los asistentes.

La entrada fue totalmente libre y gratuita, dando pie a que el número de visitantes se incrementara notablemente.

El "party" se consideró un éxito, dado que no se puede comparar con aquellos realizados en los países del norte de Europa, de los que nos encontramos todavía a años luz en estos temas. Asistieron miembros de los grupos españoles de PC anteriormente citados, Iguana y The Lords of the Bits, y también representantes de grupos nacionales de Amiga (Parabelum, Ozone y Darkness). Es de destacar la afluencia de público interesado por el tema.

No hubo competiciones pero todos los grupos asistentes mostraron sus últimas producciones y "previews" de sus futuros trabajos. También se pudieron observar los últimos lanzamientos de grupos que no estaban presentes así como la proyección de videos documentales sobre temas relacionados con la informática en todos sus aspectos.

Debido a la buena respuesta que ha obtenido Fallas Party '95 es probable que se repita la experiencia, llegando incluso a poder ser de carácter internacional en un futuro no muy lejano. Desde estas páginas esperamos manteneros informados. ■

Una demo es un programa realizado por una o varias personas que tratan de sacar el máximo partido a su computadora tanto en gráficos como en sonido, creando una "demostración" de lo que pueden llegar a hacer con sus ordenadores. ¿Quién no ha visto alguna vez Second Reality, Panic o HeartQuake? Seguramente el lector habrá tenido el placer de disfrutar con alguna de ellas.

ENTREVISTA

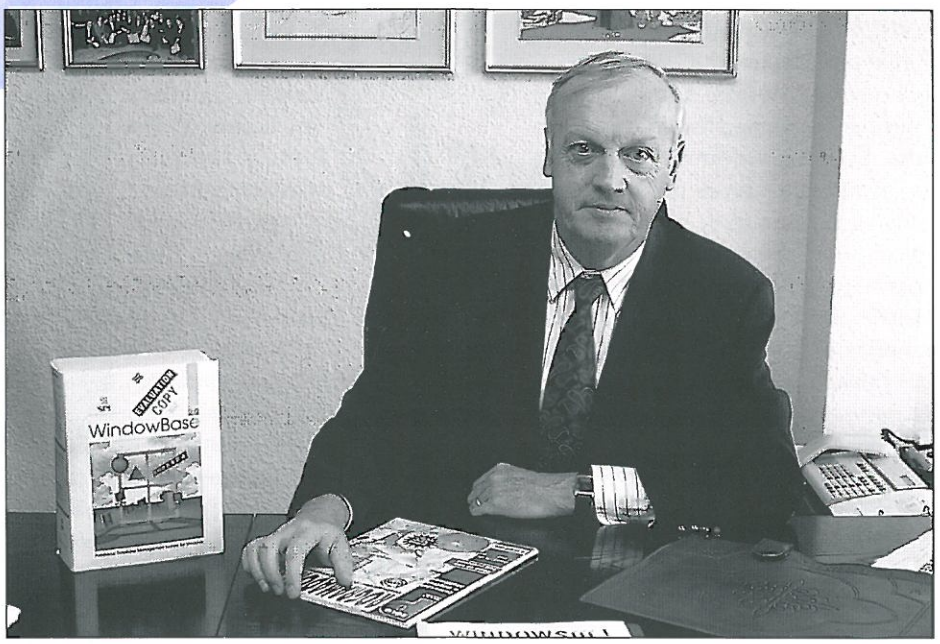
WINDOWBASE 2.0

En el mundo de las herramientas de programación y de la gestión de bases de datos hay un nuevo contendiente que aspira a hacerse con una importante cuota de mercado. Se trata de WindowBase y sus principales argumentos para conseguir sus objetivos son, la facilidad de uso, y la velocidad en el proceso de desarrollo y modificación de aplicaciones.

John Bowne, presidente y director general de Software Products International explica en esta conversación las ventajas que ofrece el nuevo producto de su compañía, y su visión sobre las posibilidades que la arquitectura cliente/servidor ofrece a los programadores.

Los beneficios que la arquitectura cliente/servidor aporta a la programación de aplicaciones, los ambiciosos planes de comercialización, la atención especial que WindowBase presta a la conectividad, y la flexibilidad del producto recién presentado son los aspectos principales de los comentarios de Bowne.

Todas estas vertientes de la herramienta le aportan los elementos necesarios para convertirla en un auténtico éxito, de la misma forma que en su día Open Access acabó convirtiéndose en el centro de la actividad creativa y comercial de SPI. Si las previsiones de Bowne y las excelencias del producto, que no se cansa de recalcar resultan ser ciertas, WindowBase será un punto de referencia obligado para todos los programadores.



La entrevista con John Bowne tuvo lugar en las oficinas de SEDYCO.

En este número, Sólo Programadores entevista s John Bowne, presidente y director general de Sotware Products International, que explica las ventajas del nuevo producto de su compañía.

¿Qué tipo de gente puede estar interesada en WindowBase? ¿Cuál es su público objetivo?

Creo que se trata de dos clases diferentes de personas. En primer lugar, el producto va destinado a programa-

dores que pretendan utilizar las bases de datos Open Access para sus aplicaciones y, por otra parte, también se orienta a aquellos que quieran crear un nuevo tipo de aplicaciones cuyo objetivo sea un aumento de la conectividad.

John Bowne interesado en un ejemplar de SÓLO PROGRAMADORES.

La arquitectura que ofrece WindowBase es lo suficientemente fuerte como para crear conexiones con XBase, cliente/servidor y todas estas cosas. Vamos a dar una nueva conectividad con el producto que está al llegar en los próximos meses.

¿Soportará todos los formatos?

Sí y lo más bonito es que las aplicaciones que se desarrollen con WindowBase funcionarán para el programador exactamente de la misma manera, con independencia de que opere en Open Access, XBase, Gupta, Oracle o DB/2.

¿Qué previsiones de venta tienen para este producto?

Si hablamos de volumen de ventas, podemos decirle que hemos vendido 350.000 copias en todo el mundo.

¿Cuántas licencias pretenden vender?

Tantas como podamos. Pero vender más de 300.000 copias en España ya sería una cifra muy importante.

¿Cuáles son las ventajas que ofrece la arquitectura cliente servidor a los usuarios de WindowBase?

Hay varias. Una es que el proceso para desarrollar una aplicación es muy, muy rápido. Como WindowBase es rápido, y además aprende por sí mismo, los seres humanos no tienen que pensar en todos los pasos necesarios para llegar desde B hasta C. En muchas aplicaciones de desarrollo, vuelves atrás y puedes darte cuenta de que has cometido un error, y tienes que volver a repetirlo todo de forma distinta, rehaciendo todo el proceso, por lo que llevar a cabo los cambios necesarios implica una buena cantidad de trabajo extra. Incluso algunas veces es más fácil volver a empezar. En WindowBase puedes ir a cualquier punto del desarrollo y hacer los cambios que sean precisos. Esto significa que cuando alguien examina la aplicación que ha creado, y llega a la conclusión de que tiene que hacer un cambio, su tarea es muy fácil ya que el producto no requiere que se de marcha atrás y se averigüe dónde está el código en el que hay que hacer los cambios. Es muy directo, y nosotros pensamos que esta es una ventaja real de la arquitectura cliente/servidor.

Por otro lado, si tienes una aplicación que, por ejemplo, trabaja contra los datos igual que Open Access y la organización decide mover algunos datos a otro sistema, DB/2, Oracle u otro, con la conectividad adecuada entre ellos, la propia aplicación los mueve directamente. No es necesario volver a programarla. Sólo hay que hacer pequeños retoques para combinar las diferentes fuentes. Ofrecemos un entorno de desarrollo consistente para todas estas aplicaciones.

¿Va a tener WindowBase un soporte técnico en España?

Sí, por supuesto. Va ha haber todo el soporte necesario. La empresa Sedyco constituye la primera línea de servicio técnico a los usuarios, y SPI será quien aporte el necesario soporte a esta firma.

¿Están trabajando ya en nuevas versiones o actualizaciones?

Sí. Obviamente, no podemos decir to-

davía de qué se trata y qué mejoras habrá, pero puedo decir que ya estamos trabajando en nuevas versiones del producto, en extensiones de WinBase y en nuevas conexiones con otras bases de datos.

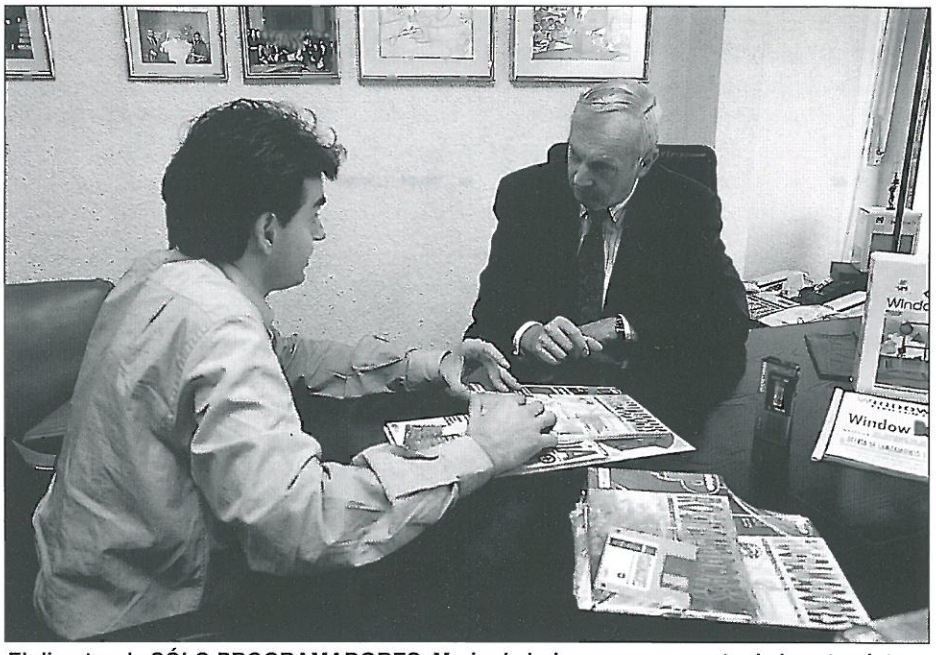
¿Cuánta gente trabaja en su compañía?

Tenemos unas veinte personas en la compañía. No es una empresa muy grande.

¿Cuáles son las ventajas que ofrece WindowBase sobre sus competidores?

Creemos que las principales son dos. Una es que es muy fácil desarrollar y modificar aplicaciones, y la otra es que es totalmente compatible con Open Access.

Si, por ejemplo, se tiene un sistema de inventario en Open Access, será necesario también un cierto número de personas instruidas para trabajar con él. Si se decide emplear WindowBase, estas mismas personas podrán trabajar simultáneamente con ambos productos mientras usan la misma base de datos, actualizándola, añadiendo datos y operando desde WindowBase.



El director de SÓLO PROGRAMADORES, Mario de Luis, en un momento de la entrevista.

¿Quizá también el precio es una ventaja?

Tenemos un precio de lanzamiento de 19.900 pesetas que creo que es muy atractivo.

¿Cuáles cree van a ser las reacciones de los programadores después de ver su producto?

Quiero decir dos cosas. La primera, que como hicimos con Open Access, hemos tomado una aproximación ligeramente diferente a las bases de datos. En segundo lugar, creo que se trata de un producto que es extremadamente fácil de usar. En la fase de pruebas "beta", muchas personas probaron el producto y al mismo tiempo también probaron otros paquetes muy populares. Esta gente nos hizo saber que lo que en otras aplicaciones les llevaba cuatro semanas, con WindowBase lo hacían en dos días. Y creemos que esto es muy importante.

Jesús Rivero, Presidente de SEDYCO, comentó a nuestra revista la labor que su empresa viene realizando con los productos de SPI.

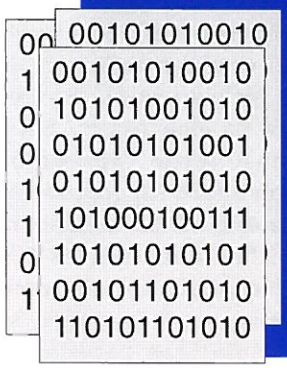
¿Cree rentable que los programadores se especialicen en el uso de una única base de datos?

Si lo que pregunta es si es bueno que los programadores se especialicen en una herramienta de programación, creo que muchos programadores se están encontrando con que usan varias aplicaciones para su trabajo, y creemos que el lenguaje de script que tenemos en este producto es muy directo, fácil de aprender y fácil de usar. Pero la cuestión es que la mayoría de los programadores no quieren llegar a ese punto, buscan simplemente el apuntar y pulsar (point & click).

Finalmente, ¿tiene algún consejo para nuestros lectores?

Pienso que hoy las posibilidades de elección son enormes, el número de aplicaciones sobre las que hay que leer es muy grande y creo que lo que deben buscar los programadores son herramientas que les hagan ser más productivos y me parece que en WindowBase tenemos un hermoso ejemplo de ello. ■

SEDYCO, S.A. (Distribuidor Exclusivo SPI. Inc.)
Teléfono, (91) 311 46 45
Fax, (91) 450 04 24



EL COPROCESADOR MATEMÁTICO

Emilio Postigo

En el capítulo anterior conocimos el funcionamiento de las instrucciones aritméticas del procesador. Éstas trabajan con números binarios enteros, y efectúan las operaciones de suma, resta, multiplicación y división.

Debido a que los cálculos habituales necesitan de operaciones más complejas y operandos en coma flotante, el programador debe implementar por su cuenta las subrutinas encargadas de cambiar el formato de los datos y las que realizan los cálculos en sí. La dificultad que ello implica es doble: por un lado es necesario codificar, al menos una vez, todos los recursos necesarios o, evidentemente, hacerse con una biblioteca que contenga las funciones necesarias. Por otro, y más importante, los programas resultan inevitablemente retrasados, debido a que una simple conversión de operandos requiere de muchas líneas de código y, por lo tanto, consume ciclos de reloj "innecesariamente", al transformar una tarea rutinaria de cálculo en un "tiempo muerto" dentro de la ejecución del programa.

EN AUXILIO DE LA C.P.U.

La solución a este problema inicial de los procesadores de la familia x86 es el empleo de un coprocesador matemático. Éste es un circuito integrado que amplía el conjunto de instrucciones del procesador, y hace posible el empleo de tipos de datos más acordes con las necesidades de cálculo habituales.

El coprocesador puede ser instalado opcionalmente en el PC, requiriéndose un modelo u otro en función del microprocesador que gobierne la placa del sistema. El primero que apareció en el mercado fue el 8087, para ser utilizado en equipos XT que dispusieran de un procesador 8086 o 8088. A medida que INTEL amplió su familia

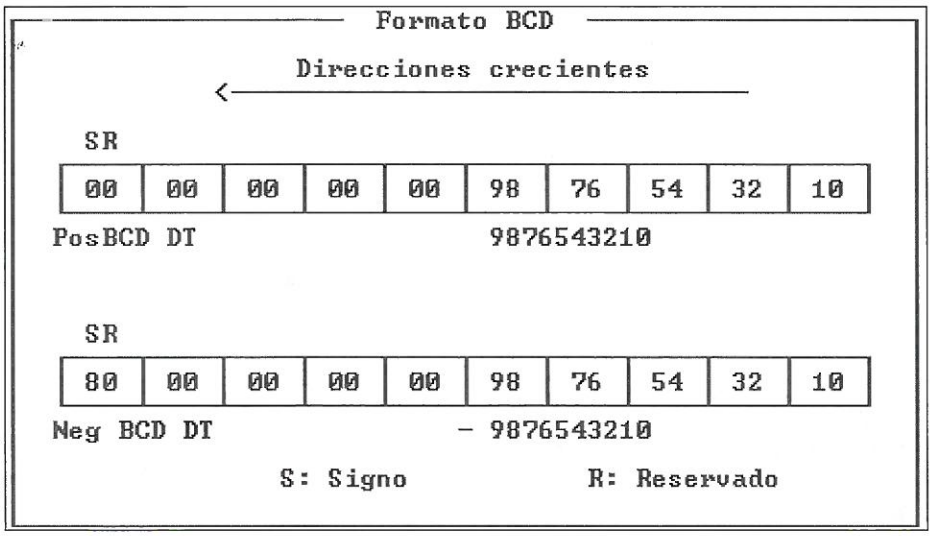


Figura 1.

Este artículo no pretende desentrañar por completo los misterios del funcionamiento de un 80x87, pero está concebido con la idea de ayudar al estudiante a "empaparse" de la mecánica de su funcionamiento y ganar soltura dentro del entorno de su manejo.

Tipo	Tamaño	Nombre
Entero binario	16 32 64	E. palabra E. corto E. largo
Entero decimal empaquetado	80	BCD
Coma flotante	32 64 80	Real corto Real largo Real temporal

Tabla 1.

con el 80286 y el 80386, surgieron los correspondientes 80287 y 80387.

El 80486 vino al mundo con el coprocesador incluido en él, por lo que añadirlo era ya innecesario. Sin embargo, más tarde apareció el 80486SX, un 80486 sin coprocesador. La razón de este hecho, extraño a primera vista, fue la de que los 80486 originales resultaban caros para el usuario medio, por lo que éste, si no necesitaba una gran capacidad de cálculo, podía disponer de las restantes prestaciones de un 80486 a un precio más asequible; en el caso de necesitar posteriormente de mayor velocidad, se podía adquirir el correspondiente 80487SX. El último miembro de la familia INTEL, el Pentium, incorpora coprocesador en todas sus versiones.

FACTORES A TENER EN CUENTA

Cuando pretende aprovechar sus recursos, el programador se topa con el hecho de que debe tener en cuenta nuevos factores a la hora de manejar el coprocesador. Por un lado éste dispone de sus propios registros, que no pueden ser direccionados de la misma manera que los del procesador. A la vez, las instrucciones del coprocesador no pueden emplear, salvo en algún caso, los registros del anterior, debiendo efectuarse el intercambio de datos entre ambos a través de la memoria convencional.

Además, los registros del coprocesador matemático difieren en longitud, formato y estructura interna de los del procesador: son de 80 bits, almacenan los datos como números en coma flotante y están organizados en forma de pila. Por este motivo se debe acceder a ellos por medio de una referencia al lugar que ocupan dentro de esa pila en un instante determinado.

Por último, aunque sólo es importante cuando se programa un 8087, se debe tener en cuenta que el sistema procesador-coprocesador está diseñado para trabajar en paralelo. En un momento determinado pueden producirse conflictos si falla la sincronización entre ambos componentes.

NUEVOS FORMATOS NUMÉRICOS

Pero no todo son sustos al acercarse al coprocesador matemático. Como ya se mencionó, éste permite emplear nuevos y potentes formatos numéricos que, junto con las instrucciones que los manipulan, hacen posible cualquier cálculo.

En las tablas 1 y 2 se muestran los diversos tipos de variables que puede manejar el coprocesador, su longitud, las directivas necesarias para definirlos y el rango de representación de cada uno de ellos. Estos formatos son empleados por el coprocesador para intercambiar información con la memoria principal. Para efectuar las operaciones necesarias éste trabaja sólo con datos de tipo REAL10, disponiéndose así del máximo rango de representación, y de una precisión más que aceptable, en la mayoría de los cálculos.

Directivas	Rango
WORD, DW DWORD, DD QWORD, DQ	-32768 → 32767 -2147483648 → 2147483647 -9.2E18 → 9.2e18
DT	-9.99E17 → 9.99E17
REAL4 REAL8 REAL10	-10E37 → 10E37 -10E307 → 10E307 -10E4931 → 10E4931

Tabla 2.

```
[+/-] parte entera•[parte fraccionaria] [E] [[+/-]exponente]
```

Figura 2.

Los datos binarios no plantean ningún problema, ya que se definen e inicializan de la forma acostumbrada. Sin embargo, antes de comenzar con los componentes del coprocesador matemático se deben remarcar algunas peculiaridades de los formatos BCD y REALx.

BCD

En primer lugar, para definir números BCD sólo puede emplearse la directiva DT señalada arriba. TBYTE, en principio equivalente, tiene un efecto distinto al deseado.

Como se mostró en el capítulo anterior, un número en base diez se almacena en formato decimal empaquetado cuando se emplea un byte para almacenar dos dígitos de ese número. Por ejemplo, el número 1234 debe quedar como 34 12 al ser guardado en memoria con este formato. Al utilizarse la directiva DT para definir una variable BCD inicializada con el valor anterior, el resultado de NumeroBCD DT 1234 es una serie de diez posiciones de memoria con los valores 00 y 10 en las dos menos significativas y cero en el resto. El programa ensamblador comprende nuestro deseo y almacena el dato de forma correcta. Sin embargo, si se escribe NumeroBCD TBYTE 1234, el ensamblador actuará de la forma habitual y transformará el dato a binario antes de almacenarlo. El resultado en este caso será D2 y 04 en las dos posiciones menos significativas y cero en el resto.

Para aclarar este punto, en la figura 1 se muestra el formato de un número BCD en memoria. Obsérvese que de los diez bytes se emplean nueve para

almacenar el módulo del número y uno, el más significativo, para guardar el signo.

COMA FLOTANTE

Las variables de este tipo deben ser inicializadas con el formato mostrado en la figura 2. Por ejemplo, a la variable NumeroREAL4, de 32 bits, se le puede asignar el número 738,98765 mediante NumeroREAL4 REAL4 0.73898765E3. Téngase presente que un número admite infinitas representaciones en coma flotante, por lo que NumeroREAL4 REAL4 +73898.765E-2 o NumeroREAL4 REAL4 73898765E-5 serían formas equivalentes de obtener el mismo resultado.

Cuando este tipo de dato se almacena en memoria, su aspecto final depende del convenio seguido por el programa ensamblador o compilador. En la figura 3 se muestra el formato interno de

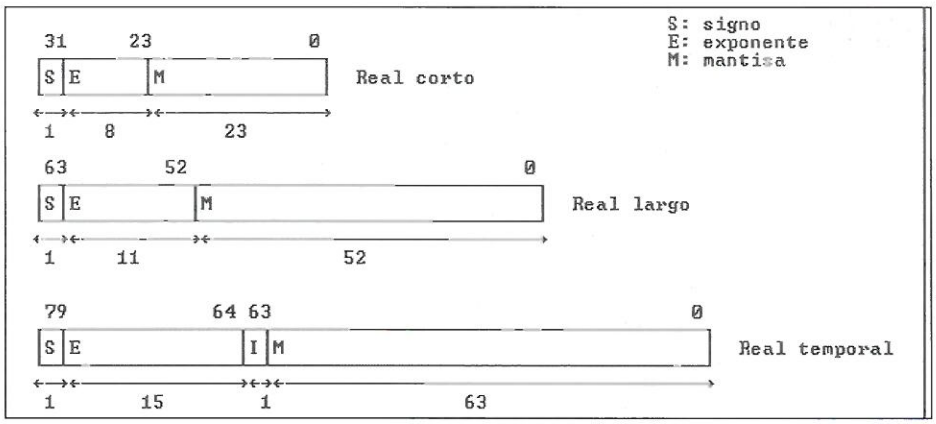


Figura 3.

los tipos REAL4, REAL8 y REAL10 según la norma IEEE 754. Como se puede observar, para cualquiera de ellos se tiene un bit de signo, unos cuantos destinados a almacenar un exponente y el resto la mantisa. El bit de signo es 0 cuando el número es positivo y 1 en caso contrario; en los bits del exponente se almacena en exceso a $2^{(N-1)}-1$ la primera potencia de 2 mayor que el número con que se trabaja y, por último, la mantisa normalizada se almacena en complemento a 2. En los tipos REAL4 y REAL8 se supone que el dígito más significativo de ésta es 1, y sólo se almacenan los que van a continuación de él. En el caso de datos tipo REAL10 esta suposición no se hace y el dígito más significativo se almacena en el bit 63. El lector que desee saber más acerca de este tipo de datos puede acudir a los libros que se citan en la bibliografía.

ARQUITECTURA DEL COPROCESADOR

Una vez conocidos los tipos de datos con los que trabaja el coprocesador, es necesario echar un vistazo a los registros que lo componen y saber, aunque sea de forma esquemática, cuál es su finalidad.

Para llevar a cabo su tarea el coprocesador dispone de trece registros: ocho registros de datos y cinco registros de control.

REGISTROS DE DATOS

Poseen 80 bits de longitud y en ellos se cargan los operandos con los que trabajan las instrucciones del coprocesador, siempre en formato REAL10 e independientemente del tipo de variable de la que se hayan extraído. Estos registros se denominan R0, R1, R2, ... R7, y tienen asignada una posición precisa dentro del sistema. Sin embargo, dado que constituyen una estructura que va a ser manejada en forma de pila, las instrucciones los van a referenciar con respecto a la posición que ocupan con respecto a la cima de ésta en un momento determinado.

El sistema posee una variable puntero interna denominada ST (de Stack-Top pointer), que se emplea para almacenar el número del registro físico Rx que debe ser considerado "en lo alto" de la pila. Este registro se referencia como ST(0) o simplemente ST, y el resto como ST(1), ST(2),... y

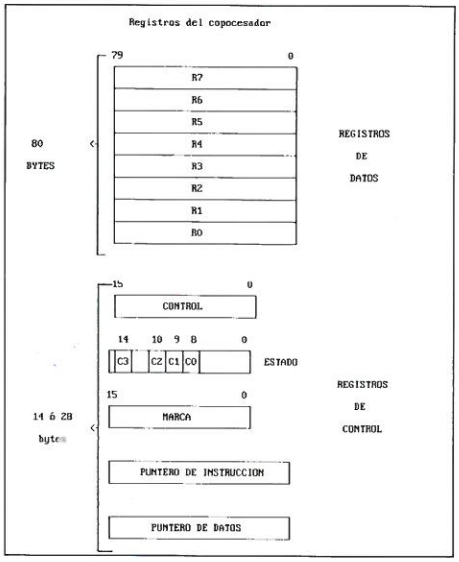


Figura 4.

ST(7). Esto significa que no hay ningún elemento "privilegiado" dentro de los registros de datos del coprocesador. Por ejemplo, si ST almacena el valor 4, ello se debe interpretar como que el registro R4 ha de ser accedido mediante el operando ST(0), R5 con ST(1), R6 con ST(2) y R7 con ST(3), encontrándose el resto "vacíos".

Entender y visualizar este modo de funcionamiento constituye el principal problema cuando se aborda por primera vez el estudio de este componente del sistema. Por este motivo, los ejemplos del capítulo están orientados a familiarizarse con esta nueva situación.

REGISTROS DE CONTROL

De los cinco registros de control, tres de ellos constan de 16 bits y los dos restantes tienen una longitud variable. Su finalidad es la de almacenar información diversa sobre el estado del coprocesador y seleccionar diversos modos de funcionamiento del mismo.

Estos registros son la palabra de control (control word), la palabra de estado (status word), la palabra marca (tag word), un puntero de instrucción (instruction pointer) y un puntero de datos u operandos (operand pointer).

La palabra de control permite seleccionar, modificando los bits apropiados, el método de redondeo, el método de manejo del "número" infinito,

el número de bits de precisión de la mantisa y las excepciones numéricas (división entre 0, entre infinito, etc) que el coprocesador debe tratar.

La palabra de estado permite que el programa obtenga información sobre la situación actual del coprocesador, como por ejemplo las excepciones numéricas que están enmascaradas, qué registro físico, de los de datos, es el primero de la pila, y si éstos contienen datos o están "vacíos", es decir, no contienen ningún operando activo. También posee ciertos bits (C0, C1, C2 y C3) que se actualizan tras comparar dos operandos y que permiten, de forma similar a los FLAGS del registro de banderas, tomar decisiones después de una comparación.

La palabra marca informa sobre cada registro de datos, señalando si su contenido es cero, infinito, válido o está "vacío".

Los punteros de instrucción y de datos almacenan la dirección física de la instrucción más reciente ejecutada por el coprocesador y la de los datos de esta última, si los tenía.

Instrucción	Constante en ST(0)
FLD1	+1.0
FLDZ	+0.0
FLDPI	PI
FLDL2E	Log2(e)
FLDL2T	Log2(10)
FLDLG2	Log10(2)
FLDLNE	Loge(2)

Figura 5.

El número de bytes ocupados por estos registros depende del modo de funcionamiento del procesador, real o protegido. En el primer caso se necesitan 14 bytes para almacenar su información, mientras que en el segundo son precisos 28.

Dado que el objetivo de este artículo es el de permitir al lector utilizar de forma inmediata las posibilidades de cálculo del coprocesador, no se entrará en detalles acerca de los registros de control del mismo. Se incidirá sin embargo en los registros de datos y en aquellos bits de la palabra de estado empleados en comparaciones. En las obras mencionadas en la bibliografía puede encontrarse toda la información necesaria para ampliar convenientemente los conocimientos sobre la estructura de estos registros y su funcionamiento. La figura 4 esquematiza estos registros y muestra la posición de C0, C1, C2 y C3 dentro de la palabra de estado.

Los registros de datos del coprocesador están organizados en forma de pila

INSTRUCCIONES DEL COPROCESADOR

Las instrucciones del coprocesador se clasifican en instrucciones de carga de variables, de carga de constantes, de almacenamiento de datos, aritméticas (suma, resta, división, etc), trascendentes (seno, coseno, logaritmos, etc), de comparación y de control del coprocesador. Estas instrucciones se distinguen de las del procesador por comenzar siempre con la letra 'F', y pueden manejar dos, uno o ningún operando.

Evidentemente, un estudio exhaustivo de cada una no tiene demasiado sentido, por lo que nos limitaremos a presentar las imprescindibles para ilustrar el modo de funcionamiento del coprocesador.

CARGA

Las instrucciones de carga introducen en la cima de la pila el operando indicado en las mismas. Este puede ser un registro ST(i) del propio coprocesador o bien uno del tipo de los mostrados en la tabla 1. El operando será convertido a real temporal (REAL10), y los cálculos pertinentes serán efectuados en este formato.

En total hay tres instrucciones de carga: FLD, FILD y FBLD. La primera permite depositar en ST el contenido de otro registro de datos o bien de una variable REALx. La segunda introduce en la pila el valor de una variable entera binaria y la última hace lo mismo con un dato tipo BCD.

En general, una 'I' detrás de la 'F' indica que la instrucción que sea trabaja con enteros, mientras que una 'B' hace referencia a números BCD. Al llegar aquí es importante recordar que no es posible cargar directamente en los registros de datos, de forma directa, los valores de los registros del procesador. El intercambio de datos entre éste y el coprocesador debe efectuarse a través de variables de la memoria principal.

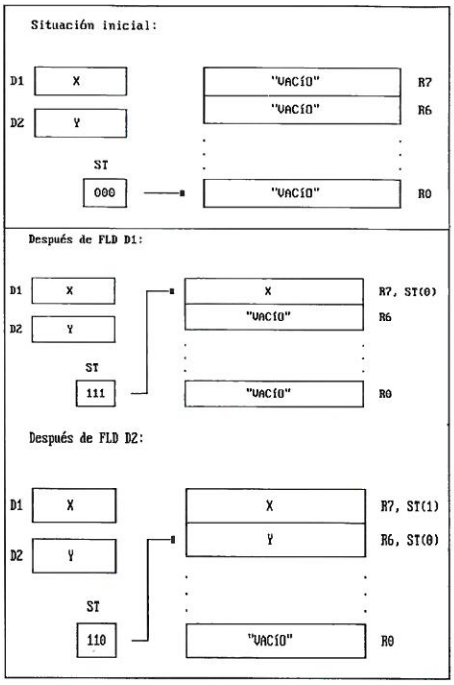


Figura 6.

Tampoco se puede introducir con estas instrucciones una constante cualquiera en la cima de la pila. Sólo algunas ya predeterminadas pueden ser depositadas en los registros de datos de una vez, y para ello se dispone de las instrucciones listadas en la figura 5. Como se puede apreciar, se trata de aquellas cuyo uso es fundamental a la hora de realizar cálculos.

Ahora se efectuará, a modo de ejemplo, el seguimiento de algunas instrucciones de carga. A la vez el contenido de la figura 6 va a ilustrar lo que ocurre dentro de la pila de registros con cada instrucción. Se supone que al comienzo de la ejecución de estas instrucciones todos los registros están libres, y que el puntero ST almacena el valor 0. En este momento, salvo con

instrucciones de carga, el coprocesador no debe manejar ningún registro de la pila. Si lo hace se activará, si no está enmascarada, una excepción de operación inválida. Tras esta aclaración, comencemos con el ejemplo.

Sean D1 y D2 variables tipo REAL4. Al ejecutarse la instrucción FLD D1 el valor de ST se decrementa. Como su valor era 0 y ST tiene un tamaño de tres bits, pasa a almacenar el valor 111 binario, es decir, el número 7. Esto quiere decir que R7 se ha convertido en la cima de la pila y que es cargado con el valor de D1 convertido a formato temporal. A continuación se ejecuta FLD D2 y ST toma el valor 6, almacenándose en él el contenido de D2. En este momento R6 constituye la cima de la pila, ST(0), y R7 debe ser referenciado, en caso de que se desee trabajar con su contenido, como ST(1).

"Congelemos" temporalmente esta situación y empleémosla como punto de partida para otro grupo de instrucciones.

INSTRUCCIONES DE ALMACENAMIENTO

Su función es depositar el valor actualmente situado en la cima de la pila en el operando especificado, convirtiéndolo al formato correspondiente.

Este grupo consta de dos tipos de instrucciones: por un lado FST y FIST, que almacenan el valor de ST en las variables oportunas (registros del coprocesador, REALx o enteras binarias), y por otro FSTP, FISTP y FBSTP, que efectúan la misma labor con el añadido de que, además, eliminan el valor correspondiente de la pila. Podemos ver esto con claridad si, como se ve en la figura 7, se prosigue con el seguimiento que quedó pendiente. Si en este momento se ejecuta la instrucción FST D3, esta variable toma el valor almacenado en ST(0), el contenido de la variable D2, quedando inalterados los registros de datos. Sin embargo, al ejecutarse FSTP D1 el valor de ST(0) pasa a la variable D1 y s el puntero ST se decrementa, siendo ahora la cima de la pila el registro R7. Por último, la instrucción FSTP D2 descarga sobre este campo el contenido del último registro ocupado y lo libera. El resultado final ha sido el de que las variables D1 y D2 han intercambiado sus contenidos, copiándose además el valor inicial de D2 en D3.

LAS RUTINAS EJEMPLO

Para mostrar más instrucciones del coprocesador y verlas actuar dentro de un programa se dispone de las subrutinas CalcRegresion, CalcMedia y CalcDesvTip. Mediante su uso combinado es posible obtener el coeficiente de correlación, la pendiente b y el corte a con el eje OY, de la recta que mejor se ajusta a una serie de puntos determinada.

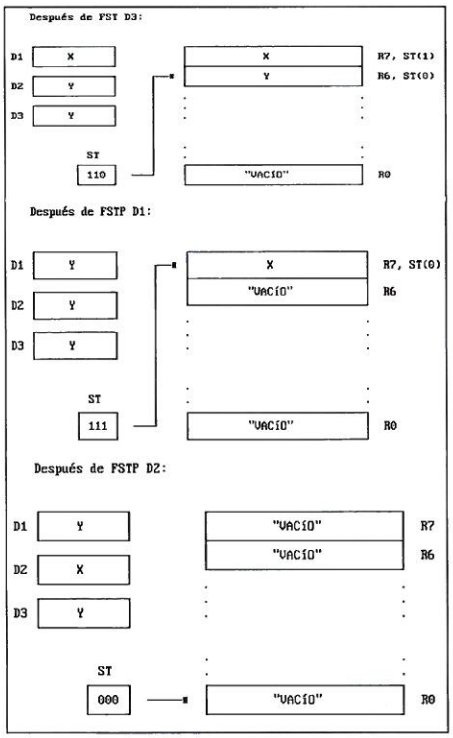


Figura 7.

La figura central es CalcRegresion, que está diseñada para ser llamada desde un programa ensamblador o C con modelo de memoria TINY o SMALL. Esta subrutina trabaja a partir de los desplazamientos de dos matrices, con un tamaño de elemento de 4 bytes (datos DWORD o REAL4) cada uno, llamados respectivamente OffsetInicioX y OffsetInicioY. Además requiere la longitud de estas matrices (Cantidad) y el desplazamiento de comienzo de la estructura de datos que va a almacenar el resultado de los cálculos(OffsetFinal). Esta subrutina se encarga de obtener los valores de 'r','a' y 'b' a partir de las fórmulas mostradas en la figura 8. Para hacer más claro el proceso se apoya en CalcMedia y CalcDesvTip, que efectúan los cálculos de las medias y desviaciones típicas de los valores contenidos en los arrays de partida, de manera que CalcRegresion sólo deba ocuparse de las fórmulas de "alto nivel".

Para seguir el código de los procedimientos no es necesario, claro está, saber de dónde salen esas fórmulas. Basta con entender qué es la media de una serie de números y que la desviación típica de esa misma serie de números es la raíz cuadrada de la diferencia (media de los cuadrados) - (cuadrado de la media).

A grandes rasgos, CalcRegresion comienza obteniendo la media y desviación típica de la variable X. Dado que no se puede seguir trabajando si la desviación es cero (se tendría una línea vertical o un sólo punto, lo que no es una situación aceptable), se pregunta por su valor y, en caso de que sea nulo se retorna al programa devolviendo el valor -2 en el coeficiente de correlación. Al no poder tomar éste un valor más pequeño que -1, el programa llamador sabrá que existe un problema con los datos iniciales.

En el caso de que la desviación típica de X no se anule, se calculan la media y la desviación típica de la variable Y, y se efectúa el mismo tipo de comprobación. Si la desviación de Y es cero, se tiene una recta de pendiente nula, lo que es completamente válido, y se devuelve un valor de 1 para el coeficiente de regresión, el valor de 'a' resultante del cálculo y 0 como pendiente de la recta.

Por último, si la desviación típica de Y no es cero, se realizan los cálculos necesarios y se obtienen 'r', 'a' y 'b', almacenándolos en la estructura que los espera en el módulo principal. El listado 1 muestra la subrutina encargada de efectuar la tarea principal.

SEGUIMIENTO DEL PROGRAMA

Una vez efectuado el reconocimiento del terreno, repitamos el proceso centrándonos en los detalles, con el fin de conocer qué es lo que hace cada instrucción. A la vez se hará hincapié en

aquellos aspectos característicos de la programación del coprocesador que vayan apareciendo.

DEVOLUCIÓN DE VALORES

En primer lugar obsérvese que después de la llamada a la subrutina CalcMedia se tiene la instrucción FSTP MediaX. Esto significa que CalcMedia ha depositado en el registro ST(0) del coprocesador el resultado de sus cálculos, pudiendo el módulo llamador recuperarla de diversas formas. En este caso se emplea la instrucción antes mencionada, que deposita el contenido de ST(0) en una variable tipo REAL4 y decrementa en 1 ST. Este método es cómodo por varios motivos, pero el más importante es que es el programa principal el que decide el formato final del valor devuelto.

La subrutina CalcDesvTip también devuelve un valor de esta manera. Sin embargo el programa principal no libera la pila después de almacenarla en una variable, sino que deja el valor correspondiente en ST(0).

FASES DE UNA COMPARACIÓN

De esta forma la siguiente instrucción, FTST, comparará ST(0) con 0 y depositará el resultado en los bits C0, C1, C2 y C3 en la palabra de estado del coprocesador.

Como se puede ver en la figura 9, el contenido de ST(0) es 0 si C3 vale 1 después de la instrucción FTST. Ahora bien ¿Cómo se puede saber cuánto vale esta bandera? Sigamos.

La instrucción FSTSW Estado, introduce en esa variable entera de dos bytes de longitud el contenido de la palabra de estado del coprocesador. A continuación se mueve el valor de la variable al registro AX y, con la instrucción SAHF, se introducen ciertos bits en el registro de banderas, de forma que el que procedía de C3 se almacena en la bandera ZF. Observando el contenido de esta bandera se puede saber si ST es 0 o no. Esto puede llevar a preguntarnos si el coprocesador es capaz de efectuar comparaciones de dos operandos cualesquiera, y la respuesta es afirmativa.

C3	C2	C1	C0	Significado
0	0	?	0	ST > 0
0	0	?	1	ST < 0
1	0	?	0	ST = 0

Figura 9.

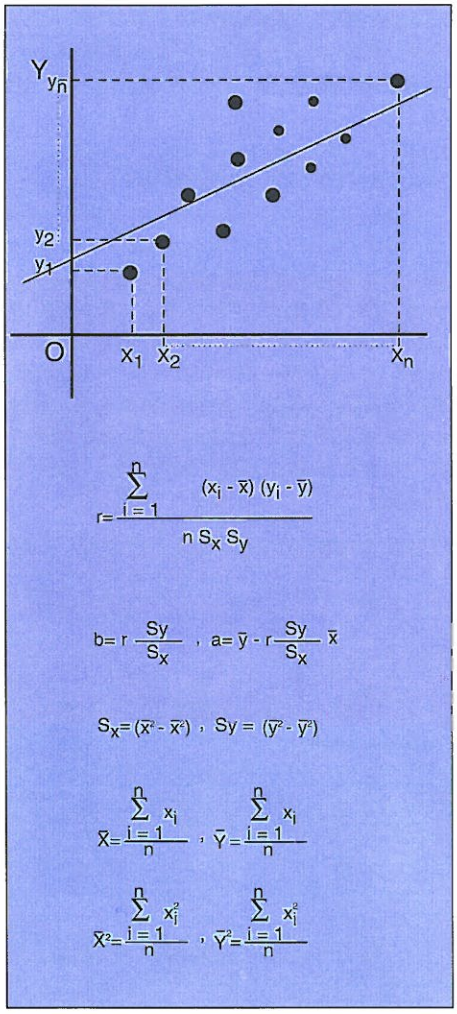


Figura 8.

Para ello dispone de las instrucciones FCOM, FCOMP, FICOM y FICOMP, que permiten comparar un registro de datos, un número en coma flotante o un entero con el valor almacenado en ST(0). La figura 10 muestra cómo ha de interpretarse el valor de C0, C1, C2 y C3 después de la comparación.

Y, un momento, ¿Qué pinta en medio de todo esto la instrucción FWAIT?

EL TEMA DE LA COORDINACIÓN

Como se dijo al principio del artículo, procesador y coprocesador son dos elementos que funcionan en paralelo. Esto quiere decir que pueden trabajar simultáneamente, cada uno con sus correspondientes instrucciones. Sin embargo, dado que se comunican entre sí mediante la memoria principal, cuando uno de ellos accede a una variable que puede estar siendo modificada por el otro, el programa ensamblador (no es necesario que el programador se ocupe de ello, salvo en algunas situaciones con el 8087) introduce una instrucción que le indica al procesador la obligación de esperar a que el coprocesador finalice su tarea. Esa instrucción es FWAIT o WAIT, que es otro mnemónico válido para la misma orden. En este caso se ha introducido explícitamente en el archivo fuente para remarcar su importancia, aunque, como ya se ha dicho, el ensamblador lo hubiera hecho de todas formas.

Siguiendo de nuevo con el ejemplo, si el contenido de ST(0) era 0, el programa debía devolver el valor -2. Éste se consigue de la forma que se puede apreciar al final del archivo fuente. Se carga en los registros ST(0) y ST(1) la constante 1 y se suman con FADDP st(1), st, ya que esta instrucción suma ST(0) sobre ST(1) y a continuación incrementa ST, con lo que la posición ST(1) de antes de la suma pasa a ser ST(0) después de ésta. Por lo tanto, ahora mismo se tiene en la cima de la pila la constante 2. Con la instrucción FCHS se cambia de signo su valor y con FSTP se almacena en la variable oportuna del módulo principal, liberándose la pila totalmente.

BIBLIOGRAFÍA

Microsoft Macro Assembler Programmer's Guide, V 6.0 y sucesivas, Microsoft Corporation
Microsoft Macro Assembler Reference, Microsoft Corporation
Strauss, Edmund: 80286, Arquitectura y sistemas. EDICIONES ANAYA MULTIMEDIA, Madrid, 1988.
E.Alcalde, M.García, S.Peñuelas: Informática básica. McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S.A.
Scheid-Di Constanzo: Métodos Numéricos. McGRAW-HILL/INTERAMERICANA DE MÉXICO, S.A.
Angulo-Funke:386 y 486-Microprocesadores avanzados de 32 bits. EDITORIAL PARANINFO, S.A.

Si la desviación típica de X no es cero, el programa calcula la de Y. Si la de esta variable es cero, el programa carga en la pila de registros la constante 1, el valor del campo MediaY y la constante 0, enviándolos a la estructura

LISTADO 1

```
CalcRegresion PROTO NEAR C par1:WORD,
par2:WORD, par3:WORD, par4:WORD
CalcMedia PROTO NEAR C par1:WORD,
par2:WORD
CalcDesvTip PROTO NEAR C par1:WORD,
par2:WORD, par3:WORD

.MODEL SMALL, C
.CODE

CalcRegresion PROC NEAR C USES bx cx di si,
OffsetInicioX:WORD, OffsetInicioY:WORD,
Cantidad:WORD, OffsetFinal:WORD,

        LOCAL DesvTipicaX:REAL4,
              DesvTipicaY:REAL4,
              MediaX:REAL4,
              MediaY:REAL4,
              ProductoXY:REAL4,
              Correlacion:REAL4,
                    A:REAL4,
                    B:REAL4,
              Estado:WORD


  mov  di,          OffsetFinal

  INVOKE CalcMedia, OffsetInicioX, Cantidad
  fstp  MediaX

  INVOKE CalcDesvTip, OffsetInicioX, Cantidad,
ADDR MediaX
  fst  DesvTipicaX

  ftst
  fstsw Estado
  fwait
  mov  ax,          Estado
  sahf

  .IF !ZERO?

    INVOKE CalcMedia, OffsetInicioY, Cantidad
    fstp  MediaY

   INVOKE CalcDesvTip, OffsetInicioY, Cantidad,
ADDR MediaY
    fst  DesvTipicaY

    ftst
    fstsw Estado
    fwait
    mov  ax,          Estado
    sahf

    .IF !ZERO?

        fldz
        fstp  ProductoXY
        mov  bx, OffsetInicioX
        mov  si, OffsetInicioY
        mov  cx, Cantidad
    @@:
        fld  DWORD PTR [bx]
        fsub  MediaX
        fld  DWORD PTR [si]
        fsub  MediaY

        fmul
        fadd  ProductoXY
        fstp  ProductoXY
        add  bx, 0004h
        add  si, 0004h
        loop  @B

        fld  ProductoXY
        fidiv Cantidad
        fdiv  DesvTipicaX
        fdiv  DesvTipicaY
        fst  Correlacion

        fmul  DesvTipicaY
        fdiv  DesvTipicaX
        fst  B
        fchs
        fmul  MediaX
        fadd  MediaY
        fstp  A

        fld  Correlacion
        fstp  DWORD PTR [di]
        fld  A
        fstp  DWORD PTR [di+4]
        fld  B
        fstp  DWORD PTR [di+8]

    .ELSE

      fld1
      fstp  DWORD PTR [di]
      fld  MediaY
      fstp  DWORD PTR [di+4]
      fldz
      fstp  DWORD PTR [di+8]

    .ENDIF
  .ELSE

    fld1
    fld1
    faddp st(1), st
    fchs
    fstp  DWORD PTR [di]

  .ENDIF


  ret

CalcRegresion ENDP

END
```

apuntada por DI. En el caso de que la desviación de Y no se anulara, el programa entra en la parte "fuerte" del mismo. En primer lugar, se carga en ST(0) la constante 0 y se vuelca sobre ProductoXY, que queda así inicializado a 0. Después, mediante un bucle cuyo comienzo viene dado por la etiqueta '@@:', se calcula el numerador de la fórmula que proporciona el valor de 'r'. Las cuatro primeras instrucciones tienen por objeto depositar en ST(0) y ST(1) las diferencias (Y - media de Y) y (X - media de X). La instrucción FSUB, cuando sólo se especifica un operando, resta al registro ST(0) el valor del operando especificado en la instrucción es tomado como dato en punto flotante, dejando el resultado almacenado en ST(0). La siguiente instrucción del bucle, FMUL, multiplica ST(0) por ST(1), dejando el resultado en ST(1) y sacando de la pila el ST(0) anterior, con lo que el resultado queda en el ST(0) actual. A continuación, el producto de las diferencias se acumula sobre la variable ProductoXY y, cuando el bucle finaliza, ésta contiene el numerador deseado, que se va dividiendo posteriormente entre los valores de Cantidad, DesvTipicaX y DesvTipicaY. Con 'r' ya calculado, se obtienen 'a' y 'b', se almacenan en la estructura destino del módulo principal y el programa finaliza.

C3	C2	C1	C0	Significado
0	0	?	0	ST > op
0	0	?	1	ST < op
1	0	?	0	ST = op

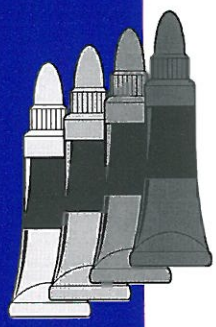
PARA TERMINAR... DE EMPEZAR

Llegados a este punto damos por concluido el capítulo. La finalidad de éste no era más, que la de presentar el coprocesador matemático y sus características básicas a aquellas personas que, sin haber tenido contacto con él y con sus recursos, deseaban poner manos a la obra cuanto antes.

Evidentemente se han quedado en el tintero gran cantidad de cuestiones, pero lo habitual a la hora de abordar el estudio de este elemento del sistema no es entender qué hace una instrucción determinada, sino cómo y con qué. En la práctica totalidad de los libros sobre lenguaje ensamblador hay una buena cantidad de páginas dedicadas al coprocesador, su estructura interna y sus instrucciones, pero el neófito suele tener dificultades a la hora de acoplar las conclusiones obtenidas en el estudio del procesador con este nuevo "ambiente hostil". Si estas páginas han servido para que el lector esté en condiciones de comenzar a comprender las características del coprocesador, habrán cumplido su objetivo. ■

UN FORMATO PARA WINDOWS, EL BMP

Agustín Guillén



Hay un formato gráfico que se encuentra en cualquier aplicación Windows, y éste es el formato BMP. Su definición, su creación y su interpretación está "servida" en el propio sistema operativo y esto facilita enormemente su inclusión en un desarrollo.

El formato BMP es el utilizado por defecto en los sistemas operativos gráficos más extendidos, como son Windows y OS/2 (sobre todo en el primero). Esto es debido a que la propia API del sistema contiene todos los procedimientos necesarios para su manejo y visualización.

El formato BMP (BitMaP) permite almacenar ficheros de 1, 4, 8 o 24 bits de profundidad de color; o lo que es lo mismo, imágenes de 2, 16, 256 o 16.777.216 colores. También puede guardar la imagen sin comprimir o con compresión RLE4 o RLE8.

Se dice que se trata de un formato DIB (Device Independent Bitmap o bitmap independiente del dispositivo) al guardarse la información de la imagen en un formato capaz de representarse en cualquier dispositivo de salida. Ésto se debe a su nacimiento dentro de un entorno gráfico, capaz de ejecutarse en cualquier configuración gráfica y de manejar impresoras de todo tipo.

Los ficheros BMP de Windows y de OS/2 son prácticamente iguales

ESTRUCTURA

Un fichero BMP se compone de las siguientes partes: una cabecera de fichero, una cabecera del bitmap propiamente dicho, una tabla de colores, y los bytes que definirán a la imagen.

Para definir las estructuras se utilizan las definiciones habituales en Windows:

BYTE	Unsigned char.
UINT	Unsigned int.
WORD	Unsigned short.
DWORD	Unsigned long.
LONG	Long.

Su formato detallado es el siguiente:

CABECERA

La cabecera contiene información sobre el tipo, el tamaño y el diseño del fichero BMP.

```
typedef                    struct
tagBITMAPFILEHEADER {
        UINT               bfType;
        WORD  bfSize;
        INT                bfReserved1;
        UINT               bfReserved2;
        DWORDbfOffBits;
} BITMAPFILEHEADER;
```

bfType Indica el tipo del fichero. Debe almacenar la cadena "BM".

BfSize Tamaño del fichero en bytes.

BfReserved1 Reservado, debe figurar un 0.

BfReserved2 Reservado, debe figurar un 0.

BfOffBits ***Offset*** desde la estructura de cabecera

BITMAPFILEHEADER a los datos del bitmap en el fichero.

INFORMACIÓN DEL BITMAP

Contiene la descripción de la imagen, así como los colores que la componen.

```
typedef struct tagBITMAPINFO {
BITMAPINFOHEADER b
miHeader;
```

```
    RGBQUAD bmiColors[1];
} BITMAPINFO;
```

Como se puede observar se compone de dos subestructuras internas:

BITMAPINFOHEADER

Estructura que contiene la información sobre las dimensiones, compresión, número de colores, resolución y demás parámetros de la imagen.

```
typedef                    struct
tagBITMAPINFOHEADER {
        DWORD biSize;
        LONG            biWidth;
        LONG            biHeight;
        WORD    biPlanes;
        WORD    biBitCount;
        DWORD biCompression;
        DWORD biSizeImage;
        LONG
biXPelsPerMeter;
        LONG
biYPelsPerMeter;
    DWORD           biClrUsed;
    DWORD           biClrImportant;
    } BITMAPINFOHEADER;
```

biSize Especifica el número de bytes que ocupa la estructura BITMAPINFOHEADER.

biWidth Anchura del bitmap en pixels.

biHeight Altura del bitmap en pixels.

biPlanes Número de planos en el dispositivo de salida. Debe estar a 1.

biBitCount Número de bits por pixel. Puede ser 1, 4, 8 o 24.

biCompression Tipo de compresión del bitmap. Puede ser alguno de los siguientes valores:

BI_RGB(0) Sin compresión.

BI_RLE8(1) Compresión RLE (run-length encoded) para bitmaps de 8 bits por pixel. Este formato se compone de 2 bytes consistentes en un byte contador y de un byte con el índice del color.

BI_RLE4(2) Compresión RLE (run-length encoded) para bitmaps de 4 bits por pixel.

BiSizeImage Tamaño de la imagen, en bytes. Si se trata de un fichero sin compresión (BI_RGB), este campo se puede quedar a 0.

BiXPelsPerMeter Resolución horizontal, en pixels por metro, del dispositivo destino para el bitmap. Este valor se utiliza para seleccionar entre un grupo de imágenes, a una determinada que mejor se adapte al dispositivo en cuestión.

BiYPelsPerMeter Resolución vertical, en pixels por metro, del dispositivo destino para el bitmap.

BiClrUsed Indica el número de colores indexados en la tabla de color que son utilizados en la imagen. Si este campo está a 0, el bitmap utiliza el máximo número de colores correspondiente al valor indicado en el campo biBitCount. Si este campo no es 0, indica el número de colores que el driver gráfico usará si el campo biBitCount es menor de 24. Si biBitCount es 24, biClrUsed indica el tamaño de la tabla de color de referencia utilizada para optimizar las prestaciones de las paletas de Windows.

BiClrImportant Número de colores indexados que son considerados importantes para la visualización del bitmap. Si el campo está a 0, todos los colores son importantes.

```
C:\MIO\ARTICUL\FORGRAF4\DISCO>infobmp.exe ajedrez.bmp
INFORMACION DEL FICHERO BMP
Tamaño del fichero          : 65078
Offset a la imagen          : 1078

Tamaño de BITMAPINFOHEADER: 40
Ancho de la imagen          : 320
Alto de la imagen           : 200
Planos del destino          : 1
Planos de color             : 8
Tipo de compresión          : 0
Tamaño de la imagen         : 64000
Resolución horizontal       : 0
Resolución vertical         : 0
Número de colores usados    : 0
Núm.de colores importantes: 0
```

Información suministrada por el programa ejemplo INFOBMP. EXE.

RGBQUAD

Esta estructura describe un color en el formato RGB.

```
typedef struct tagRGBQUAD {
        BYTE            rgbBlue;
        BYTE            rgbGreen;
        BYTE            rgbRed;
        BYTE            rgbReserved;
    } RGBQUAD;
```

rgbBlue Intensidad azul del color.
rgbGreen Intensidad verde del color.
rgbRed Intensidad roja del color.
rgbReserved No utilizada, debe ser 0.

DATOS

Son los datos propiamente dichos de la imagen. Cada línea del bitmap

Permite dos tipos de compresión diferentes RLE8 y RLE4, además de un modo raw

Un fichero BMP consta de 4 partes: cabecera de fichero, del bitmap, tabla de colores y bytes de imagen

debe ser redondeada a un LONG, rellenando con ceros el espacio sobrante. El origen de la imagen es la esquina inferior izquierda.

Como se ha dicho anteriormente, se pueden comprimir imágenes con 4 y 8 bits por pixel.

Como varían parcialmente en sus definiciones se van a describir por separado:

RLE8

El modo absoluto se activa cuando el primer byte es 0 y el segundo se encuentra entre 0x03 y 0xFF. En este modo se copiará íntegramente el número de pixels indicado en el segundo byte a partir del siguiente byte. Cada grupo de datos codificados en este modo deben ser par o alineados a word.

RLE4

En el modo codificado, el primer byte de la pareja contiene el número de pixels a ser dibujado utilizando los valores contenidos en el segundo byte.

El segundo byte contiene dos valores diferentes de color o pixels: el primero en su nibble (grupo de 4 bits o medio byte) alto y el segundo en el bajo.

Estos se van repitiendo hasta llegar al número de valores indicado en el primer byte.

En el modo absoluto, el primer byte es 0 y el segundo indica el número de valores a copiar a partir del siguiente byte, siempre partiéndolos en nibbles.

Cada grupo en el modo absoluto debe ser par (word boundary) y los códigos de escape definidos anteriormente también son válidos en este modo de compresión.

Como ejemplo para comprender la compresión utilizada, a continuación se lista un volcado de una parte de los datos de la imagen de un fichero BMP, comprimidos utilizando RLE8:

```
02 AA 04 02 00 04 24 38 53 88 03
21 00 00
00 02 02 03 08 67 00 01
```

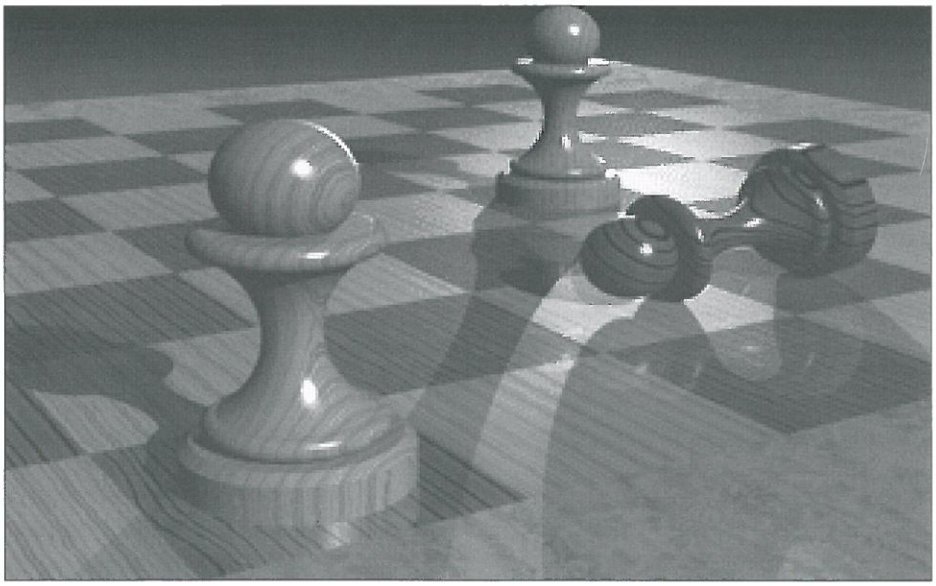
Al descomprimirse se obtendrían los siguientes datos:

```
AA AA
02 02 02 02
24 38 53 88
21 21 21
Fin de línea
Salto 2 posiciones a la derecha y 3
abajo
67 67 67 67 67 67 67 67
Fin de la imagen
```

El formato BMP (BitMaP) permite almacenar ficheros de 1, 4, 8 o 24 bits de profundidad de color

La compresión puede utilizar dos modos diferentes: codificado y absoluto; estos modos se pueden entremezclar dentro de un mismo bitmap.

El modo codificado consiste en dos bytes que indican el número pixels consecutivos en el primer valor y el valor a repetir en el segundo.



Si el primer byte es 0, se trata de un código de escape que contiene unos de los siguientes valores en el siguiente byte:

0 fin de línea.
1 fin del bitmap.
2 salto a otra zona. Los dos bytes siguientes indican el offset del desplazamiento al siguiente pixel desde la posición actual.

EJEMPLO PRÁCTICO

Para ilustrar el acceso a estas estructuras, se ha desarrollado un sencillo programa que muestra en pantalla la información contenida en un fichero BMP, leyendo la cabecera del fichero, la cabecera del bitmap, la paleta de colores y los datos de la imagen. Como los componentes de cada color en un fichero BMP están en el orden BGR (RGB en el hardware VGA), sus valores varían entre 0 y 255 (entre 0 y 64 en VGA) e intercala un byte no utilizado para cada color; se incluye una rutina que convierte la paleta del fichero a una utilizable por el sistema gráfico del PC. También tiene en cuenta que los datos de un fichero BMP están almacenados de abajo a arriba y de izquierda a derecha.

Para simplificar el programa, sólo muestra imágenes que cumplan los siguientes requisitos: 320 x 200 pixels de tamaño, 256 colores (8 bits por plano) y sin ningún tipo de compresión. ■

IMPLEMENTACIÓN

M. A. Cazorla

EL análisis de requisitos suele estar asignado al analista, que debe plantearse todas la preguntas que pueda imaginar sobre la aplicación a desarrollar y darlas respuesta. Por ejemplo tiene que pensar si se posee toda la información, si no se ha olvidado algún detalle, si el problema se ha subdividido hasta hacerlo factible, etc.

En el programa que estamos creando, en principio no se ha olvidado ningún detalle (el lector seguro que sí habrá encontrado algunos, pero una vez que se adquiere cierta experiencia programando, hay ciertos aspectos que se obvian por costumbre).

Sin embargo, existen "lagunas" en cómo se hacen ciertas cosas, como por ejemplo leer de teclado cualquier tecla, hacer que se muevan por la pantalla la raqueta y la pelota o conseguir una buena interactividad.

Es decir, hay que documentarse en aquello que no se domine al 200%.

DOCUMENTACIÓN

Un programador debe disponer de toda la información relativa a la máquina sobre la que desarrolle por lo que es conveniente poseer libros sobre la arquitectura física de todos los componentes del ordenador, del sistema operativo y disponer de varios compiladores y diversos lenguajes de programación, ya que no hay nada que lo haga todo.

Un buen ejemplo son las Fichas de Programación de Sólo Programadores, que reúnen datos e informaciones adquiridas a lo largo del tiempo por experimentados programadores.

Para conseguir hacer el Frontón se requiere, por ejemplo, una tabla ASCII para reconocer la tecla pulsada y poder representar convenientemente la pelota y la raqueta.

Los detalles que conviene conocer más a fondo son los siguientes:

- Cómo se leen los "cursores" del teclado.
- Cómo se representan los objetos en la pantalla.
- Cómo se consigue la interactividad.

La calidad de la interactividad está en función de lo intuitiva, cómoda y rápida que sea la forma de realizarse.

LECTURA DE LOS CURSORES

La función Read de Turbo Pascal, la herramienta empleada en la codificación de este "juego", permite obtener cualquier tecla perteneciente a los caracteres ASCII, sin embargo el teclado posee otras teclas como son por ejemplo, las de función, tabulador, ... y los cursores.

Para que el juego sea fácil de controlar, hay que acceder al empleo de las teclas correspondientes a los cursores.

¿Cómo se hace?. La respuesta es inmediata, se mira en un libro. Existen muchos dedicados a la programación de los trucos del PC, en nuestro caso

Un proyecto informático se divide en tres etapas bien diferenciadas: Definición, desarrollo y mantenimiento. En el capítulo anterior se completó la primera, pero antes de pasar a teclear, hay que realizar un nuevo análisis de los requisitos y problemas que pueden surgir durante la implementación.

LISTADO 1

```
TYPE
TipoTecla = ( nula, f1, f2, f3, f4, f5, f6, f7, f8, f9,
f10, enter, tab, shiftab, bksp, arriba, abajo, dere-
cha, izquierda, del, ins, home, esc, endk, texto,
numero, espacio, pgup, pgdn);

Procedure tecla
 ( var tecla_f: tipotecla; var caracter: char);
VAR ok:boolean;
   c:char;
BEGIN
 ok:=false;
 c:=readkey;
 if (c=#0) and keypressed
 then begin
     ok:=true;
     c:=readkey;
     end;
 if ok then
   case ord(c) of
    15:tecla_f:=shiftab;
    72:tecla_f:=arriba;
    80:tecla_f:=abajo;
    82:tecla_f:=ins;
    75:tecla_f:=izquierda;
    77:tecla_f:=derecha;
    73:tecla_f:=pgup;
    81:tecla_f:=pgdn;
    71:tecla_f:=home;
    79:tecla_f:=endk;
    83:tecla_f:=del;
    82:tecla_f:=ins;
    59:tecla_f:=f1;
    60:tecla_f:=f2;
    61:tecla_f:=f3;
    62:tecla_f:=f4;
    63:tecla_f:=f5;
    64:tecla_f:=f6;
    65:tecla_f:=f7;
    66:tecla_f:=f8;
    67:tecla_f:=f9;
    68:tecla_f:=f10;
    end
 else
  begin
  caracter:=c;
  case ord(c) of
   8:tecla_f:=bksp;
   9:tecla_f:=tab;
   13:tecla_f:=enter;
   27:tecla_f:=esc;
   32:tecla_f:=espacio;
   33..44,47,58..254:tecla_f:=texto;
   45,46,48..57:tecla_f:=numero;
  end;
 END;
```

Obtención de la tecla pulsada.

emplearemos "Programer's guide to the IBM PC", edición del 1985 de Peter Norton (Microsoft Press). En el capítulo 6 de este libro dedicado al teclado encontramos la respuesta:

Cuando se pulsa una tecla correspondiente a los caracteres ASCII el código recibido varía entre 1 y 255. Si el leído es el 0, entonces una tecla especial ha sido pulsada, por lo que hay que volver a leer del teclado para saber cual es.

En la tabla de este libro hay identificadas 97 teclas especiales, pero en la actualidad hay más como por ejemplo las teclas F11 y F12, que seguramente corresponden a los códigos 69 y 70 respectivamente.

En el listado 1 se puede ver de forma práctica cómo obtener cualquier tecla.

El procedimiento TECLA averigua que tecla se pulsó, informando en el primer parámetro el tipo o nombre especial de la tecla y en el segundo el carácter ASCII correspondiente.

La definición del tipo TipoTecla simplifica bastante la codificación y el desarrollo del programa, ya que permite emplear símbolos de objetos reales.

Se ha procurado que los nombres de las teclas sean lo más representativas posibles. Como detalle curioso, la tecla END (FIN) se tuvo que definir como ENDK, al producirse un error de compilación. En Pascal, la palabra END está reservada como significador de fin de sentencia, procedimiento o programa.

DIBUJO DE LOS OBJETOS

La técnica está basada en los dibujos animados. Sencillamente se trata en ir dibujando el objeto en la nueva posición, habiendo borrado previamente la posición anterior.

Para ser más concretos vamos a "pintar" lo que serán las definitivas pelota y raqueta. De nuevo hay que recurrir a la indispensable tabla ASCII. Como sobre gustos no hay nada escrito, será el autor de este artículo el que decida con mayor o menor acierto el aspecto de estos elementos.

Para la pelota conviene que sea un carácter de aspecto redondo, y el elegido es el correspondiente al código 79, O.

La raqueta, siguiendo el estilo de juegos similares se puede representar como una barra horizontal. El tamaño permitirá seleccionar el nivel de dificultad. El carácter elegido esta vez es el 205, =.

El analista debe plantearse todas la preguntas que pueda imaginar sobre la aplicación a desarrollar y darlas respuesta

LISTADO 2

```
Begin

...

Repeat
 gotoxy(PosPelotaX, PosPelotaY);
 writeln('O');
 gotoxy(PosRaquetaX, PosRaquetaY);
 writeln(raqueta[NDificultad]);

 PosAntPelotaX:= PosPelotaX;
 PosAntPelotaY:= PosPelotaY;
 PosAntRaquetaX:= PosRaquetaX;
 PosAntRaquetaY:= PosRaquetaY;

 Tecla( TeclaPulsada, caracter);

 (*
    Realización de los cálculos pertinentes en ba-
se a
    la tecla pulsada y de la nueva posición de
    pelota y raqueta
 *)

 gotoxy(PosAntPelotaX, PosAntPelotaY);
 writeln('O');
 gotoxy(PosAntRaquetaX, PosAntRaquetaY);
 writeln(raqueta[NDificultad]);
Until TeclaPulsada=ESC;

...

END.
```

Algoritmo para dibujar.

CUADRO 2

Un tipo enumerado puede considerarse como un conjunto de objetos que pueden ser empleados como valores. De echo suponen un nuevo tipo de datos como puede ser el tipo CHAR, o BOOLEAN.

Ejemplo: Definición del tipo predefinido BOOLEAN.

```
TYPE  Boolean= (False, True);
```

Una vez definido pueden crearse variables de este tipo, por ejemplo:

```
VAR correcto: Boolean;
```

Ahora la variable "correcto" podrá tomar los valores False o True.

Como curiosidad, el compilador asigna a cada valor del tipo un número, comenzando por el 1, así False valdrá 1 y True 2. Para obligar a que tomen otros índices se realiza de la siguiente forma:

```
TYPE ValoresLogicos = (falso=0, verdad=1);
```

Por esto podemos referirnos a los caracteres ASCII por su símbolo o por su código:

Símbolo	Código
O	79
=	205

Definición de tipos enumerados.

¿Cómo se pintan?. Las funciones que aporta Turbo Pascal son Gotoxy(i,j) y Writeln(n). La primera coloca el puntero de escritura en pantalla en la posición i,j. Writeln se encarga de escribir en el lugar fijado anteriormente el carácter o conjunto de caracteres.

Un programa es interactivo cuando requiere de la actuación en tiempo de ejecución del usuario

Para borrar el símbolo basta con escribir en esa misma posición el mismo carácter pero con el color del fondo.

Pero no es suficiente con pintar y borrar. Hay que saber cuando conviene hacerlo. Por ejemplo si se pinta, se borra y luego se hacen cálculos, los dibujos parpadearán considerablemente. Sin embargo, si se pintan, se hacen los cálculos y por último se borran, quedará una apariencia más continua.

INTERACTIVIDAD

Un programa es interactivo cuando requiere de la actuación en tiempo de ejecución del usuario.

La calidad de la interactividad está en función de lo intuitiva, cómoda y rápida que sea la forma de realizarse.

Ha de ser intuitiva para facilitar su aprendizaje. Para esto ha de parecerse a acciones habituales del ser humano. Aunque la raqueta no se mueve sólo de izquierda a derecha, es una buena forma de hacerlo en el juego, a parte de haberse demostrado que el empleo de las teclas "izquierda", "derecha", "arriba" y "abajo" ha dado muy buenos resultados.

La acción intuitiva no se limita al teclado, sino también a cómo suceden las cosas en la pantalla. Si la pelota se mueve tan aprisa que es imposible verla, se habrá cometido un error. No da igual tampoco que el toque de la pelota con la raqueta sea rígido. Habrá que simular un cierto retroceso o sensación de impacto a ambos objetos para causar sensación de realismo en el juego.

El juego debe ser cómodo, ya que se limitaría su uso por complejo. El número de teclas empleadas debe ser mínimo. En Frontón sólo se emplearán los cuatro cursores para gobernar la raqueta y la tecla ESC para acabar la partida.

La rapidez es otro factor importante. Si se tarda demasiado en calcular la posición de la pelota y se mueven muy despacio los objetos, nadie querrá jugar. Por otro lado, como se apuntaba anteriormente, si la pelota no se llega a ver nunca es que no se puede jugar de lo rápido que se mueve.

CUADRO 1

Unidad	Función	
Gotoxy	CRT	Coloca cursor en posición.
Keypressed	CRT	True si se pulsa una tecla.
Readkey teclado	CRT	Obtiene carácter de directamente.
Textcolor	CRT	Cambia color caracteres.
Textbackground	CRT	Cambia color de fondo.
Wherex	CRT	Obtiene posición horizontal del cursor.
Wherey	CRT	Obtiene posición vertical del cursor.
Writeln	Estándar	Escribe caracteres ASCII.

Funciones y procedimientos.

PRÓXIMO CAPÍTULO

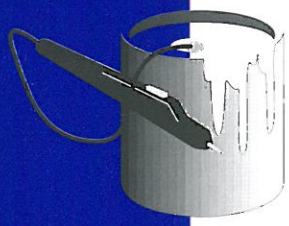
Prácticamente está hecho el programa aunque alguien crea lo contrario. Sólo falta codificar (fase de desarrollo) y un detalle muy importante: la documentación o redacción de manuales tanto para el usuario como para otro programador. ■

Las fases de un proyecto informático son definición, desarrollo y mantenimiento

Conviene documentarse en aquello que no se domine al 200%

TRATAMIENTO DIGITAL DE IMÁGENES (II)

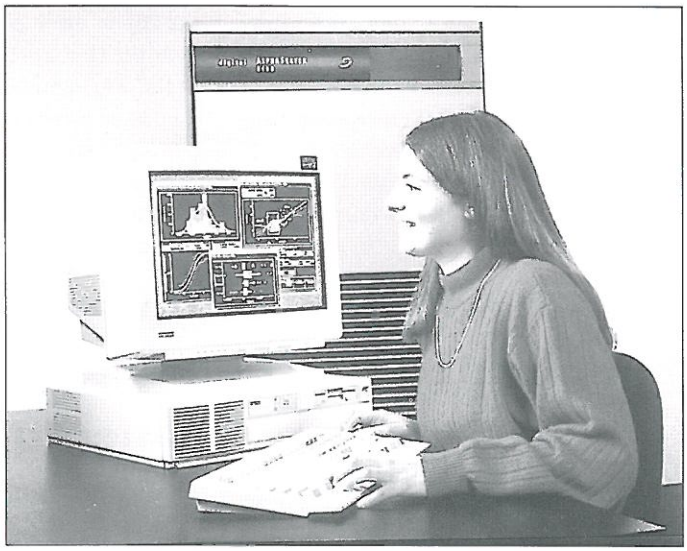
Enrique Coiras



En el artículo anterior se explicaban algunas transformaciones sencillas para el procesado digital de imágenes. En este se verán algunos métodos de realzado de imágenes, otra técnica para extraer bordes y el método del filtrado homomórfico para perturbaciones multiplicativas

MÉTODOS DE REALZADO DE IMÁGENES

Los métodos de realzado de imágenes son un conjunto de técnicas que intentan mejorar la apariencia visual de las imágenes tratadas. Estas técnicas no pretenden obtener una representación mas fiel del objeto o escena original, como ocurre con los métodos de restauración o reconstrucción de imágenes, sino simplemente una imagen que resulte más clara o subjetivamente atractiva para el ojo humano.



Una de las técnicas de realzado mas utilizadas es la mejora de contraste, que ya se vio en el artículo precedente. Otra, llamada sharpening o afilado, es la que busca aumentar la nitidez de los detalles contenidos en una imagen. Esto normalmente se consigue reforzando los bordes de las figuras contenidas en la misma. Al contrario que las técnicas de suavizado o desenfoque, que se quedaban con las frecuencias bajas de las imágenes, de lo que se trata aquí es de resaltar las frecuencias altas, es decir, los detalles y, cómo no, también el ruido.

Una técnica simple para realzar los bordes consiste en convolucionar la imagen con una máscara discreta que ofrezca buena respuesta a las altas frecuencias, como por ejemplo :

$$H = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Este método, además de ser sencillo, suele dar resultados aceptables en la mayoría de las ocasiones, aunque hay otros bastante más potentes.

Una forma muy ingeniosa para quedarse con las frecuencias altas consiste en restarle a la imagen sus frecuencias bajas, es decir, la propia imagen suavizada. Claro que, esto no debe hacerse directamente, porque las frecuencias bajas contienen la información concerniente al nivel medio de gris de la imagen, y al restar la imagen suavizada se obtendría una imagen bastante más oscura que la original.

La potencia de los ordenadores domésticos permite realizar funciones de tratamiento de imágenes que hasta hace poco estaban reservadas exclusivamente a las estaciones de trabajo y ordenadores dedicados.

Figura 1 : Una imagen y su versión afilada mediante la técnica de unsharp masking.

Así que, en la práctica, para que el nivel medio de gris no varíe, lo que se hace es sumarle a cada punto la diferencia entre su valor real y su valor suavizado, multiplicada por una cierta constante. A esta técnica se la denomina unsharp masking. Variando el valor de la constante y el grado de suavizado podemos controlar el grado de afilado de la imagen.

En la figura 1 puede verse el resultado de su aplicación. La imagen final aparece más nítida y los detalles se aprecian mejor que en la original. La rutina utilizada es la siguiente :

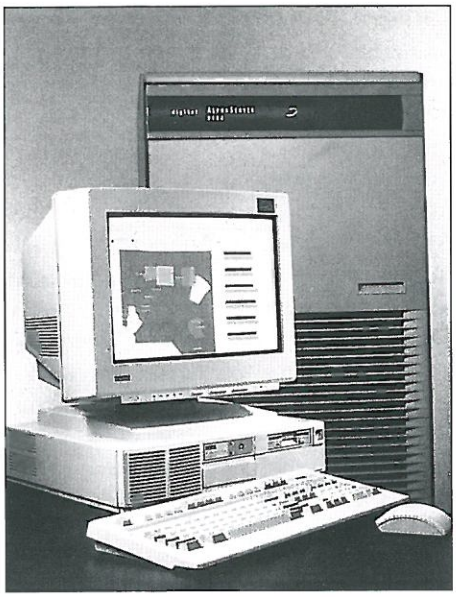
```
void unsharp_mask (imagen img, int numsoften, float k)
 {
  fimagen fimg, sfimg;
  int y, x;

  crearfimagen (&fimg, img.sizey, img.sizex);
  crearfimagen (&sfimg, img.sizey, img.sizex);

  imagen2fimagen (img, fimg);

  suavizado_media (img, numsoften);
  imagen2fimagen (img, sfimg);

  for (y = 0; y < img.sizey; y ++)
   for (x = 0; x < img.sizex; x ++)
    fimg.dat [y][x] += k * (fimg.dat [y][x] - sfimg.dat [y][x]);
  truncarf (fimg, 0.0, (float) (defnumgrises - 1));
  fimagen2imagen (fimg, img);
  destruirfimagen (&sfimg);
  destruirfimagen (&fimg);
 }
```

También se puede hacer un afilado sumándole a la imagen su propio gradiente, pero así, más que aumentar la nitidez, lo que se consigue es resaltar groseramente los bordes, ya que el gradiente no es muy nítido por sí mismo. Sin embargo, la laplaciana, o derivada del gradiente, sí es lo suficientemente fina como para que pueda conseguirse un buen afilado con ella. Al igual que ocurre con la discretización de muchas otras funciones (como el gradiente, por ejemplo) la laplaciana discreta se puede calcular de muchas formas. Una de las más utilizadas consiste en convolucionar la imagen con alguna de las máscaras 3x3 del cuadro 1. La rutina siguiente utiliza la máscara separable de cuatro vecinos para el cálculo de la laplaciana :

```
void laplacianaf (fimagen img, fimagen imgout)
 {
  int masksize = 3;

  int laplazmask [3][3] = {0, -1, 0, -1, 4, -1, 0, -1, 0};
  fimagen mask;
  int y, x;

  crearfimagen (&mask, masksize, masksize);

  for (y = 0; y < masksize; y ++)
   for (x = 0; x < masksize; x ++)
    mask.dat [y][x] = 0.125 * (float) laplazmask [y][x];

  prod_convolucionf (img, mask, imgout);

  destruirfimagen (&mask);
 }
```

Los métodos de realzado mejoran sensiblemente las imágenes

Al igual que ocurría en el unsharp masking, en el afilado con laplaciana es una constante la que nos controla el grado de realzado. Como se ve en la figura 2 los resultados del afilado con laplaciana y los del unsharp masking son bastante parecidos. A continuación está la rutina utilizada :



```
void realz_lap (imagen img, float k)
{
  fimagen fimg, flap;
  int y, x;

  crearfimagen (&fimg, img.sizey, img.sizex);
  crearfimagen (&flap, img.sizey, img.sizex);

  imagen2fimagen (img, fimg);
  laplacianaf (fimg, flap);

  for (y = 0; y < img.sizey; y ++)
    for (x = 0; x < img.sizex; x ++)
      fimg.dat [y][x] += k * flap.dat [y][x];

  truncarf (fimg, 0.0, (float) (defnumgrises - 1));
  fimagen2imagen (fimg, img);

  destruirfimagen (&flap);
  destruirfimagen (&fimg);
}
```

Existen otros algoritmos de afilado, pero son bastante más complicados que el unsharp masking y el de la laplaciana, y no ofrecen resultados mucho mejores.

La técnica de afilado busca aumentar la nitidez de las imágenes

El afilado de bordes por diferenciación estadística consiste en multiplicar el valor de cada pixel de la imagen por su desviación estándar estimada :

Imgout (x, y) = Img (x, y) * D (x, y)

donde la desviación estándar se define como :

D(x,y)=1/W^2*SUM(m=x-w;x+w)SUM(n=y-w;y+w){Img(m,n)-M(m,n)}^2

y se calcula en una vecindad de tamaño WxW alrededor de cada pixel con W = 2 * w + 1 (esto nos asegura que W sea impar). La función M (x, y) es la media estimada de la imagen de entrada en el punto (x, y) y se calcula de la siguiente manera :

M(x,y)=1/W^2*SUM(m=x-w;x+w)SUM(n=y-w;y+w){Img(m,n)}

El algoritmo de suavizado de media que apareció en el artículo pasado hace precisamente esto, así que podemos utilizarlo para calcular la media.

La imagen de salida, Imgout (x, y), tendrá más amplitud que la imagen original en aquellos pixels que se desvíen significativamente de sus vecinos, mientras que la amplitud decrecerá en los demás puntos. El proceso es análogo al del control automático de ganancia para una señal de audio.

DETECCIÓN DE BORDES MEDIANTE LAPLACIANA

Antes hemos podido utilizar la laplaciana para resaltar los contornos debido a la información que esta nos aporta sobre las frecuencias altas de la imagen. También podemos utilizar esta información para detectar los bordes de las figuras contenidas en la misma. Aunque en el artículo pasado ya se vio como podían extraerse los bordes con el gradiente, para usar la laplaciana debemos tener en cuenta algunas consideraciones.

La laplaciana es más sensible que el gradiente y destaca mucho más el ruido que aquel. Por ello, antes de extraer los bordes, es conveniente pasarle un suavizado gaussiano a la imagen, o bien utilizar máscaras grandes (5x5, 7x7...) para calcularla.

En cualquier caso, para extraer los bordes con la laplaciana no basta con calcular su módulo, como ocurría con el gradiente, ya que eso genera bordes

El unsharp masking y el afilado con laplaciana dan resultados equivalentes

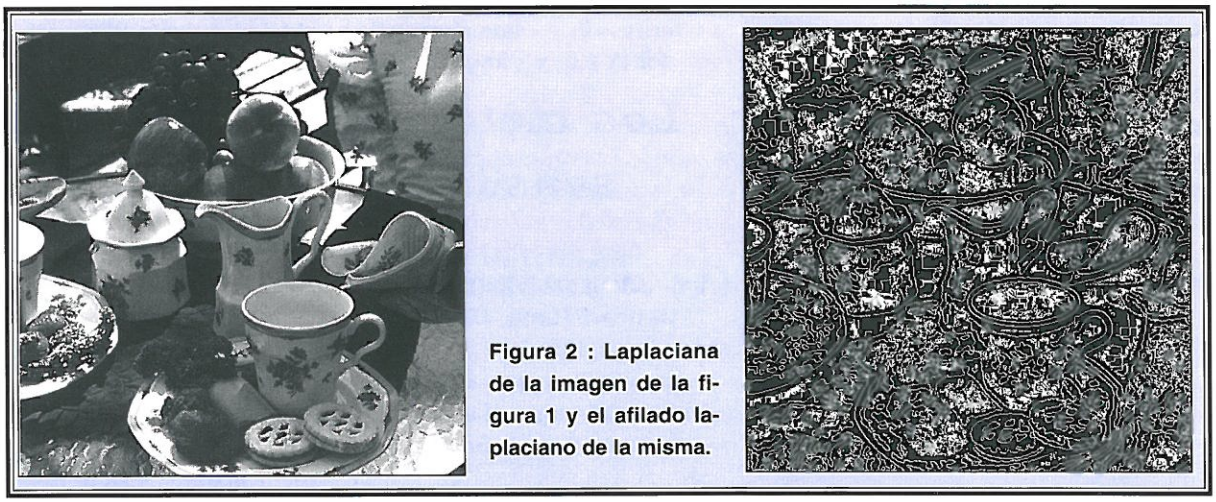
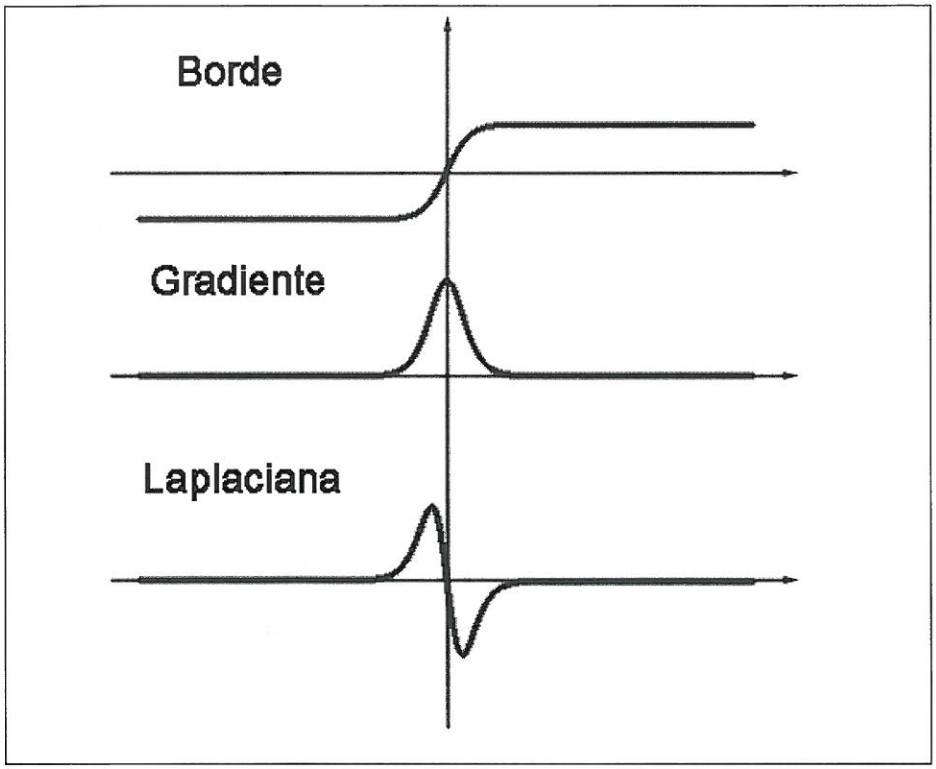


Figura 2 : Laplaciana de la imagen de la figura 1 y el afilado laplaciano de la misma.



cuadro 2 : Comparación de los comportamientos del gradiente y la laplaciana en las proximidades de un borde.

dobles (las diferencias de comportamiento entre gradiente y laplaciana pueden verse en el cuadro 2). Hay que buscar sus zero-crossings, es decir, los puntos en los que la laplaciana pasa de positivo a negativo o viceversa; y hay que hacerlo tanto para filas como para columnas. Debe tenerse en cuenta el hecho de que el pixel en el que se produce el zero-crossing no tiene porque valer cero. Así que los casos con los que nos podemos encontrar son los siguientes :

+0- , -0+ , +-* y -+*

En cualquiera de ellos debe marcarse el punto como borde (el asterisco indica cualquier valor posible). La rutina que sigue hace precisamente esto :

```
void lapzerocross (imagen img, imagen imgout)
{
  int y, x;
  fimagen fimg, lap;

   crearfimagen (&fimg, img.sizey, img.sizex)
   crearfimagen (&lap, img.sizey, img.sizex);
   imagen2fimagen (img, fimg);
   laplacianaf (fimg, lap);
   borrarimagen (imgout);

   for (y = 1; y < img.sizey - 1; y ++)
      for (x = 1; x < img.sizex - 1; x ++)
         if (((lap.dat[y][x] == 0.0) && ((lap.dat[y-1][x] * lap.dat[y+1][x] < 0.0) || (lap.dat[y][x-1] * lap.dat[y][x+1] < 0.0))) || ((lap.dat[y-1][x] * lap.dat[y][x] < 0.0) || (lap.dat[y][x-1] * lap.dat[y][x] < 0.0)))
            imgout.dat [y][x] = (unsigned char) (defnumgrises - 1);
         else
            imgout.dat [y][x] = 0;

   destruirfimagen (&lap);
   destruirfimagen (&fimg);
}
```

La extracción de bordes mediante esta técnica tiene varias características particulares. Una de ellas es que los contornos que nos aparecen son cerrados, lo cual es deseable en algunas tareas específicas, como la segmentación, que se utiliza para el reconocimiento automático de formas y siluetas. Otra particularidad es que la imagen de bordes resultante ya está binarizada, esto es, en cada punto hay o no hay borde, pero no existen valores intermedios como ocurriría con el gradiente.

Una variante más refinada de esta técnica consiste en utilizar como umbral el gradiente de la imagen para decidir en que zonas se aplica la detección de bordes mediante laplaciana. Así se obtienen bordes aun más limpios y claros.

FILTRADO HOMOMÓRFICO

El filtrado homomórfico es una técnica de mejorado bastante útil en los casos en los que la imagen fuente esta perturbada por ruido multiplicativo, interferencias o iluminación poco uniforme. El ruido multiplicativo se caracteriza por que su amplitud es proporcional a la de la señal, lo cual, en el caso de imágenes digitales, viene a significar que las partes de la imagen que son más claras se ven más afectadas por el ruido que las más oscuras. Las técnicas de eliminación de ruido que se vieron en el pasado artículo no pueden utilizarse en este caso, ya que están diseñadas para limpiar ruido aditivo (esto es, sumado a la imagen).

El unsharp masking refuerza las frecuencias altas para realzar los contornos

El filtrado homomórfico resuelve este tipo de problemas de una forma muy ingeniosa. Supongamos que la imagen de partida, Img (x, y), puede modelarse como producto de una imagen sin ningún tipo de perturbación, A (x, y), y una matriz de interferencia I (x, y). Es decir :

$$Img(x, y) = A(x, y) * I(x, y)$$

Tomando logaritmos en ambos miembros y recordando que log (a *

El filtrado homomórfico permite eliminar las perturbaciones multiplicativas

b) = log (a) + log (b) queda lo siguiente :

log (Img (x, y)) = log (A (x, y)) + log (I (x, y))

Por ello es necesario sumarle a la imagen original un valor constante cualquiera antes de extraer el logaritmo; y después, una vez calculada la exponencial, hay que restárselo a la imagen resultado. En el cuadro 3 puede verse un esquema de todo el proceso completo.

Figura 3 : Extracción de bordes según los zero-crossings de la laplaciana.

Y ahora ya es posible utilizar las técnicas de filtrado lineal de las que se disponga para reducir cuanto sea posible la aportación logarítmica de la interferencia. Una vez hecho esto no hay más que calcular la exponencial de la imagen filtrada para obtener la versión mejorada de la imagen de partida.

Hay que tener cuidado al calcular el logaritmo, ya que este presenta una singularidad en el origen.

La laplaciana también puede utilizarse para la extracción de bordes

La rutina siguiente utiliza el método del filtrado homomórfico para hacerle un unsharp masking a una imagen :

CUADRO 1

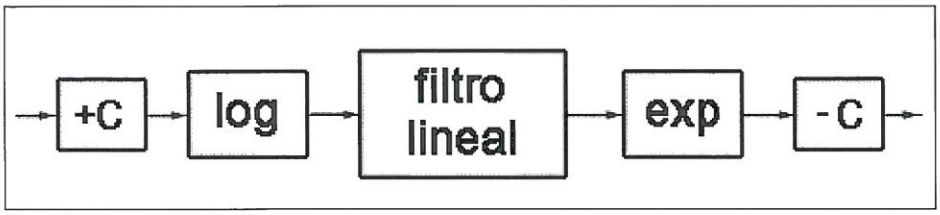
laplaciana separable de 4 vecinos

```
        [-2  1 -2]
L = 1/8 * [ 1  4  1]
        [-2  1 -2]
```

laplaciana separable de 8 vecinos

}}}

Dos de las máscaras más utilizadas para el cálculo de la laplaciana.



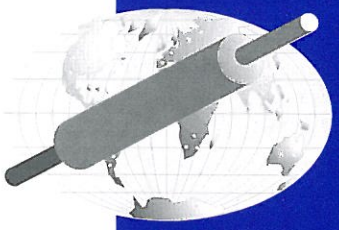
Esquema de funcionamiento del filtrado homomórfico.

```
void filtrado_h (imagen img, int n, float k)
{
  int y, x;
  fimagen logimg, fimg;

  crearfimagen (&logimg, img.sizey, img.sizex);
  crearfimagen (&fimg, img.sizey, img.sizex);

  for (y = 0; y < img.sizey; y ++)
    for (x = 0; x < img.sizex; x ++)
      logimg.dat [y][x] = (float) log ((double) img.dat [y][x] + 1.0);
  unsharp_maskf (logimg, n, k);

  borrarfimagen (fimg);
  for (y = 2 * n; y < img.sizey - 2 * n; y ++)
    for (x = 2 * n; x < img.sizex - 2 * n; x ++)
      fimg.dat [y][x] = (float) exp ((double) logimg.dat [y][x]) - 1.0;

  normalizarfimagen (fimg, 0.0, (float) (defnumgrises - 1));
  fimagen2imagen (fimg, img);

  destruirfimagen (&fimg);
  destruirfimagen (&logimg);
}
```

En lugar de la llamada al procedimiento de unsharp masking puede realizarse el filtrado que se desee : mediana, Wiener, mejora de contraste, etc. ■

RDSI. LA REVOLUCIÓN DE LAS COMUNICACIONES TELEFÓNICAS

Francisco Monteagudo



Años 60: Se encuentra la solución a un viejo problema, la perdida de calidad de sonido en las llamadas a larga distancia. La solución consistía en utilizar canales de larga distancia digitales; en estos canales la voz era digitalizada y enviada como datos numéricos, volviéndola a convertir en una señal analógica en el otro extremo de la línea (figura 1). Puesto que en los enlaces digitales la información no sufre deterioro, las llamadas continentales podían tener la misma calidad de sonido que las llamadas locales. El esquema de digitalización elegido fue tomar muestras, que en Europa eran de 8 bits y en EE.UU. de 7 bits, a una velocidad de 8000 muestras por segundo; esto significaba que estos canales debían funcionar a 64000 bits por segundo en Europa y 56000 bits por segundo en EE.UU.

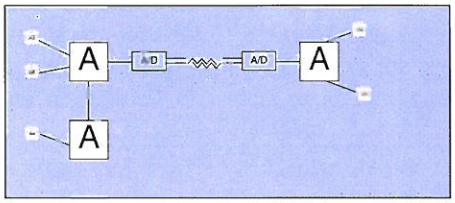


FIGURA 1: Red telefónica clásica con enlaces de larga distancia digitales.

Años 70: Las compañías telefónicas se enfrentan a un nuevo desafío; las grandes empresas están interesadas en poder interconectar sus ordenadores; para satisfacer esta nueva demanda se crean las primeras redes experimentales de transmisión de datos.

Año 1984: Asamblea general de la CCITT. Este organismo, dependiente de la ONU, tiene como función establecer los estándares técnicos utilizados en telefonía, con el fin de garantizar la compatibilidad entre los equipos de las diferentes compañías. En esta reunión se habla de los canales digitales, del imparable aumento de las comunicaciones por ordenador y de las nuevas demandas ya aparecidas o de previsible aparición (fax, videotexto, videoconferencia, televisión por cable, ...), y se toma una decisión histórica; la red telefónica mundial deberá reconvertirse en una red de transmisión de datos. El plan es que, en el siglo XXI, las típicas líneas analógicas utilizadas por los teléfonos de voz se habrán sustituido por líneas digitales capaces de ofrecer cualquier tipo de servicio, inventado o por inventar; esta nueva red se bautiza con el nombre de RDSI (Red Digital de Servicios Integrados).

La idea era muy buena, pero presentaba un problema enorme, la construcción de esta red. Si se quería que el proyecto fuera viable, la nueva RDSI debía crearse a partir de la vieja red de voz. El esquema finalmente elegido fue el de un desarrollo en dos fases; en una primera fase se sustituirían las viejas centrales de relés por nuevas centrales computerizadas, que aunque serían compatibles con los sistemas antiguos podrían ofrecer los servicios requeridos por la nueva red; paralelamente, todos los canales de comunicación (no sólo los de larga distancia) se irían reconvirtiendo en canales digitales. Esto permitiría la existencia de un período de transición durante el cual estarían entremezclados enlaces analógicos y di-

Mientras los modems se esfuerzan para ser cada vez más rápidos (ya han llegado a los 28800 bps), de una forma callada pero imparable se está introduciendo la tecnología destinada a reemplazarlos; una tecnología que funciona a una velocidad mínima de 64000 bps; el RDSI.

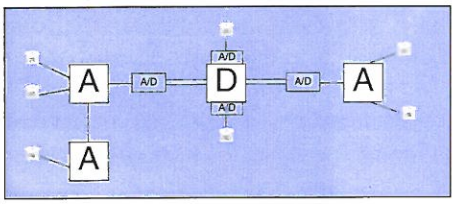


FIGURA 2: Red telefónica con enlaces analógicos y digitales entremezclados.

gitales (figura 2) y que concluiría en la RDI (Red Digital Integrada), una red en la que el único enlace analógico sería el que une el teléfono del abonado con la central (figura 3). Llegados a este

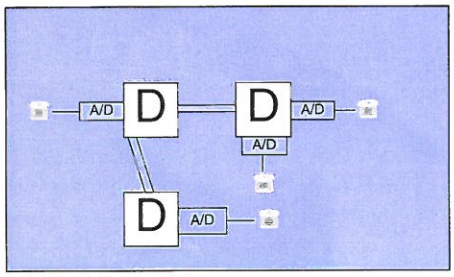


FIGURA 3: Red RDI.

punto, se entraría en la segunda fase, que consistiría en alargar los enlaces digitales hasta los abonados (figura 4); la RDSI habría nacido.

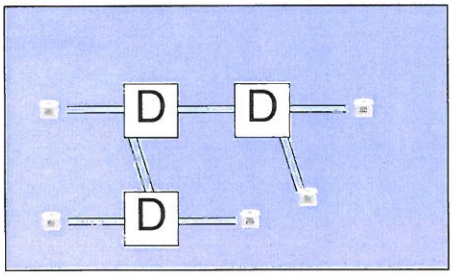


FIGURA 4: Red RDSI.

Años 90: Muchos países han completado la construcción de la RDI; puede ponerse en marcha la RDSI.

Esta es la situación actual; como puede verse en la figura 3, en el contexto de la RDI el teléfono del abonado está conectado a un conversor analógico/digital que convierte la señal eléctrica en información binaria que será transmitida a través de un canal de datos; en el otro extremo del canal, un conversor digital/analógico reconstruye la señal original. No olvidemos que en la red telefónica, el canal de voz es la unidad básica de funcionamiento; esto significa que la RDI estará formada por grupos de canales de 64 kbps en Europa y 56 kbps en EE.UU., lo que también supone que esta deberá ser la velocidad de los canales RDSI.

CANALES DE DATOS Y CANALES DE SEÑALIZACIÓN

Como ya se ha dicho, la RDSI está formada por canales de comunicación digital a 64 Kbps, pero para las comunicaciones se necesita algo más; es necesario controlar la comunicación, es necesario poder llamar y colgar. Para estas funciones de control se utiliza un canal aparte, el canal de señalización; mediante este canal, con un protocolo de mensajes, se inician y terminan las llamadas y se realizan todas las funciones típicas disponibles en las líneas telefónicas modernas (y que las líneas RDSI conservan); funciones como retención de llamada, conferencia a tres, redirección de llamada, etc.

En la terminología técnica, los canales de transmisión de datos se denominan canales B, y los canales de señalización se denominan canales D.

PROBLEMAS DE COMPATIBILIDAD

El desarrollo del RDSI plantea dos problemas, el primero es la ya mencionada necesidad de mantener la compatibilidad con los equipos telefónicos existentes, y el segundo es que, puesto que se trata de una creación muy reciente, existen muchos aspectos que todavía no están adecuadamente estandarizados.

Como se comentaba al principio de este artículo, en Europa y EE.UU. la velocidad a la que trabajan los canales digitales de voz es diferente, lo que supone que la RDSI también irá a velocidad diferente. Por si esto fuera poco, el protocolo utilizado en el canal de señalización también es diferente a cada lado del Atlántico. Esto significa que los equipos de conexión a la red europeos y americanos son diferentes e incompatibles; es decir, que antes de comprar un equipo, el usuario debe asegurarse de que funcionará con su instalación telefónica.

Para no complicar las explicaciones, este artículo se referirá únicamente a las normas RDSI europeas (conocidas como EuroISDN).

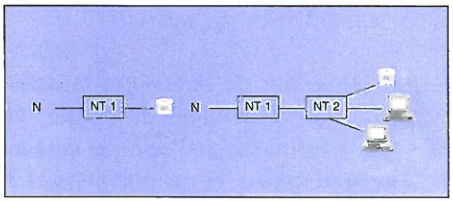


FIGURA 5: Esquema de una conexión básica.

ANATOMÍA DE UNA LÍNEA RDSI

Las compañías telefónicas ofrecen dos tipos fundamentales de líneas RDSI, las líneas básicas (BRI) y las líneas primarias (PRI). Una línea BRI consiste en un cable de dos o de cuatro hilos; dos son para la transmisión de datos y los dos hilos opcionales se utilizan para proporcionar alimentación eléctrica al terminal NT1. Sobre este cable se multiplexan dos canales B y un canal D a 16Kbps, lo que da una velocidad total de 144Kbps. Una línea PRI puede ser un cable coaxial o de fibra óptica sobre el que se multiplexan 30 canales B y un canal D a 64Kbps, lo que da una velocidad total de 1984Kbps. En el lado del abonado, como puede verse en la figura 5, la línea BRI finaliza en un

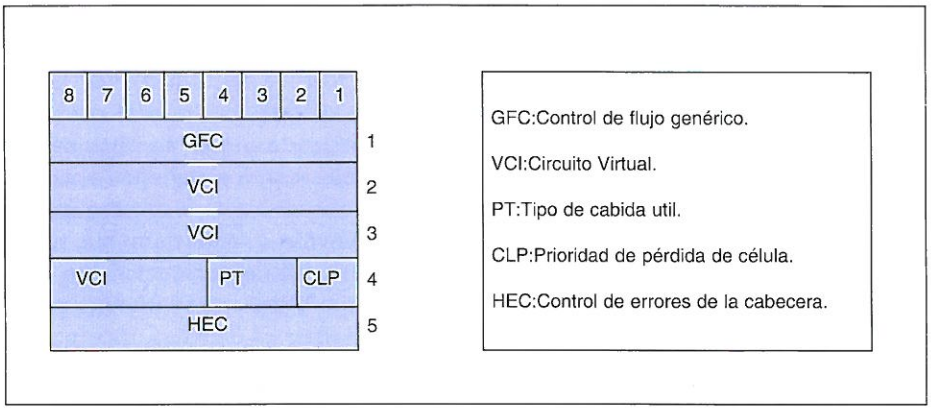


FIGURA 7: Protocolo ATM.

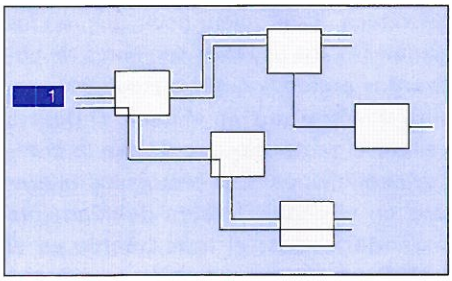


FIGURA 6: Red basada en circuitos virtuales.

terminal NT1, dispositivo que, en esencia, es un modem; este aparato tiene un terminal de salida de 4 líneas llamado bus S/T, al cual se pueden conectar los equipos terminales (teléfono/fax RDSI, ordenador, ...) o un terminal NT2, que es un multiplexor que permite tener conectados varios equipos terminales a un mismo terminal NT1. Una línea PRI, en cambio, se conecta a una central (PBX) que dispone de interfaces para la conexión de terminales NT2.

CIRCUITOS VIRTUALES

Existen múltiples técnicas para multiplexar varios canales sobre un mismo cable, y la elegida por el CCITT fue el uso de circuitos virtuales.

Un circuito virtual es una técnica que se utiliza en protocolos de comunicaciones basados en paquetes (la unidad mínima de información es un paquete de bits o bytes). Cada paquete lleva una etiqueta que identifica un camino dentro de la red, camino que indica la ruta que deben seguir estos paquetes para ir desde el ordenador origen hasta el destino. Esto se puede ver en la figura 6.

Los protocolos de circuitos virtuales disponen de un subprotocolo para abrir y cerrar dichas rutas, es decir, para "llamar" (crear una ruta que conecte nuestra máquina con otra máquina de la red) y "colgar" (eliminar esa ruta).

PRIMER NIVEL, EL PROTOCOLO ATM

El protocolo básico de la RDSI es el ATM (Asincronous Transfer Mode). En él, el paquete (celda) tiene una longitud de 53 bytes, dividida en una cabecera de 5 bytes, y un campo de datos de 48 bytes de longitud; el formato del paquete puede verse en la figura 7.

El ATM es un protocolo atípico en muchos sentidos; así, no incluye el subprotocolo para crear y eliminar los circuitos virtuales; además, un protocolo de comunicaciones corriente incluye en su cabecera una suma de chequeo, que permite detectar los errores producidos dentro del paquete durante la transmisión, y unos números de secuencia que tienen una doble función, por un lado sirven para que el receptor pueda ordenar los paquetes si estos le llegan desordenados, y por otro lado sirven como referencia para, en caso de error, poder indicarle al emisor cual ha sido el paquete defectuoso para que lo vuelva a enviar. En el protocolo ATM encontramos un campo de chequeo, pero que sólo comprueba la cabecera, por lo que el sistema es incapaz de detectar errores en el campo de datos; además, si se detecta un error en una cabecera la celda es descartada, no está previsto ningún mecanismo para recuperar las celdas con errores.

La RDSI utiliza canales diferentes para la comunicación y para el control de la línea

La razón de todas estas peculiaridades es que, en primer lugar, el ATM está concebido para ser implementado por hardware, en módulos que serán programados externamente (por eso no incluye mecanismos para abrir y cerrar circuitos virtuales); en segundo lugar, está previsto que los circuitos virtuales sean fijos, lo que significa que las celdas siempre seguirán el mismo camino a través de la red, por lo que siempre llegarán a su destino

0	0	0	0	0	0	0	0
1	D1	D1	D1	D1	D1	D1	S1
1	D1	D1	D2	D2	D2	D2	X
1	D2	D2	D2	D2	D3	D3	S3
1	D3	D3	D3	D3	D3	D3	S4
1	1	0	0	E4	E5	E6	E7 a)
1	D4	D4	D4	D4	D4	D4	S6
1	D4	D4	D5	D5	D5	D5	X
1	D5	D5	D5	D5	D6	D6	S8
1	D6	D6	D6	D6	D6	D6	S9

0	0	0	0	0	0	0	0
1	D1	D1	D2	D2	D3	D3	S1
1	D4	D4	D5	D5	D6	D6	X
1	D7	D7	D8	D8	D9	D9	S3
1	D10	D10	D11	D11	D12	D12	S4
1	1	1	0	E4	E5	E6	E7
1	D13	D13	D14	D14	D15	D15	S6
1	D16	D16	D17	D17	D18	D18	X
1	D19	D19	D20	D20	D21	D21	S8
1	D22	D22	D23	D23	D24	D24	S9

FIGURA 9: Adaptación de velocidades utilizando el V.110.

Número de octeto	Número de bit							
	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0
1	1	D1	D2	D3	D4	D5	D6	S1
2	1	D7	D8	D9	D10	D11	D12	X
3	1	D13	D14	D15	D16	D17	D18	S3
4	1	D19	D20	D21	D22	D23	D24	S4
5	1	E1	E2	E3	E4	E5	E6	E7
6	1	D25	D26	D27	D28	D29	D30	S6
7	1	D31	D32	D33	D34	D35	D36	X
8	1	D37	D38	D39	D40	D41	D42	S8
9	1	D43	D44	D45	D46	D47	D48	S9

FIGURA 8: Protocolo V.110.

en el mismo orden en el que fueron enviadas (por eso no se incluyen números de secuencia); y en tercer lugar, porque es un protocolo diseñado para ser rápido, para que el tiempo que tardan las celdas en viajar desde el emisor hasta el receptor sea mínimo, lo que significa eliminar cualquier proceso intermedio que imponga retardos; por eso no se incluye ningún mecanismo de corrección de errores.

ENCAPSULADO DE INFORMACIÓN BINARIA

El protocolo ATM está concebido para encapsular información procedente de canales binarios síncronos utilizando el esquema de multiplexación por tiempo; esto quiere decir que el controlador lee el estado del canal X veces por segundo (siendo X la velocidad de transmisión de ese canal) y los bits obtenidos los va guardando en el campo de datos de una celda ATM; cuando esta celda se ha rellenado, se envía. En el lado receptor, se leen los bits de la celda y se van escribiendo en el canal de salida a la misma velocidad X a la que fueron leídos; si una celda se pierde, se considera que todos sus bits valían uno. Nótese que, desde el punto de vista de los dispositivos, el canal binario es un registro de 1 bit que es escrito por el emisor y leído por el receptor. Si se dispone de varios canales binarios (lo normal en la RDSI), este proceso se aplica en cada canal por separado, y cada celda ATM contendrá información procedente de un único canal.

De lo dicho se desprende que los canales ATM transmiten información binaria pura, sin ningún tipo de formato ni protección contra errores; exactamente de la misma forma como la transmitiría un cable serie que conectara directamente el emisor y el receptor; así, en la comunicación RDSI, el protocolo ATM proporciona el nivel físico.

SEGUNDO NIVEL, CANALES B Y D

Como se ha visto, dentro de una línea RDSI tanto los canales B como el canal D son líneas binarias síncronas, la diferencia entre ellos (aparte de la velocidad) es que, mientras los canales B pueden conectarse a cualquier otro sistema (a cualquier otro número de teléfono), el canal D está permanentemente conectado al ordenador de la central telefónica. Esto quiere decir que, en los canales B, los usuarios son libres de utilizar los protocolos que deseen (en teoría), mientras que en el canal D deberá utilizar el protocolo que decida la compañía telefónica. Los protocolos utilizados en el canal D están debidamente estandarizados; el más básico es el LAP-D (que es un subconjunto del protocolo HDLC), el encargado de la detección y corrección de los errores de transmisión; encima de éste, encontramos el Q.931, que es el protocolo de señalización propiamente dicho, y el X.25, que permite usar este canal también para la comunicación entre usuarios.

TERCER NIVEL, PROTOCOLOS DENTRO DEL CANAL B

En principio, como se ha dicho, nada impide que dos usuarios de RDSI conectados mediante un canal B se pongan de acuerdo para utilizar el protocolo que deseen, pero el RDSI es un servicio creado por las compañías telefónicas para permitir una comunicación abierta y fluida, por lo que se han definido algunos como estándar. Existen dos que se puede considerar como los mas básicos; uno es el A-law, que no es un protocolo de comunicaciones, sino el método empleado para digitalizar la señal de voz; este fue el esquema que se eligió en su día para codificar la señal de voz y enviarla por los canales digitales; este sistema conserva en la RDSI; este es el único protocolo bajo el cual un terminal RDSI puede comunicarse con un teléfono ordinario; el segundo es el BONDING, que es un procedimiento que permite fusionar un grupo de canales B para formar un único canal de comunicación a alta velocidad; así, en el enlace básico se puede hacer un bonding de los dos canales B disponibles para formar un canal de 128 Kbps; eso sí, no hay que olvidar que la compañía telefónica tarifica canales B, por lo que en este ejemplo se tendría que pagar por dos llamadas.

El CAPI proporciona el interfaz entre los programas de aplicación y el RDSI

La gama es bastante más amplia; el protocolo de nivel inferior es el V.110; es un protocolo que permite reducir la velocidad de un canal B, haciendo posible la conexión a otras redes de menor velocidad (por ejemplo, la RDSI americana); encima de éste, se encuentra el HDLC, que es el encargado de proporcionar la detección y corrección de errores. Encima de estos se sitúa el protocolo X.75, que es el que realmente gestiona las comunicaciones por el canal.

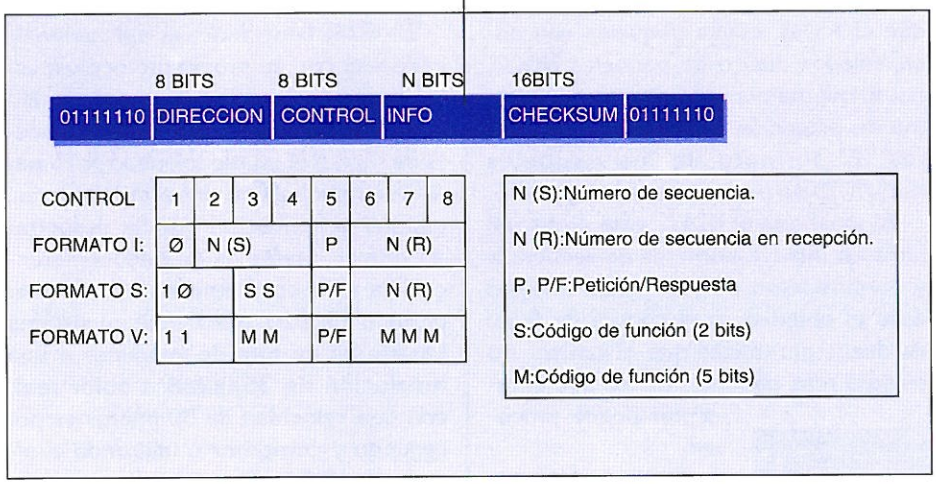


FIGURA 10: Protocolo HDLC.

V.110, LA COMPATIBILIDAD OBLIGADA

Imaginemos que una compañía telefónica desea que los usuarios de la RDSI puedan hacer conectar con modems analógicos conectados a la red convencional; ¿cómo podría hacerlo? la solución es simple; introducir una pasarela que convierta los bits procedentes del canal B en una señal de modem. Sin embargo, queda un problema pendiente; los modems funcionan, como máximo, a 28.800 bps. La solución a este problema es simple, reducir la velocidad del canal B para igualarla a la velocidad del modem; sin embargo, existe un grave problema; no se puede cambiar la velocidad de un canal RDSI. Para resolverlo se creó el V.110, que es un protocolo que encapsula un pequeño número de bits de datos procedentes del canal de baja velocidad (en este caso, el modem) dentro de tramas de 80 bits de longitud, organizadas en forma de paquetes de 10 bytes, que se envían por el canal de alta velocidad.

Como se ve en la figura 8, esta trama puede almacenar hasta 48 bits de datos; en la figura 9 se pueden ver algunos de los encapsulados utilizados. En esta gráfica también se puede ver que la conversión no es a 64Kbps; esto se debe a que la conversión se hace en dos pasos, en el primero se convierte desde el canal de baja velocidad a un canal intermedio de 8, 16 o 32Kbps, y en el segundo paso se convierte a 64Kbps. En esta figura también se puede ver que cada bit se almacena repetido; esto se hace como mecanismo de protección contra los errores en la transmisión. Los equipos terminales deben disponer de un módulo capaz de empaquetar y desempaquetar dichas tramas, entregando (y recogiendo) los bits de datos al nivel superior (normalmente, el HDLC).

El protocolo ATM es el corazón de las redes RDSI

HDLC, EL NIVEL DE ENLACE

El protocolo HDLC se diseñó para proporcionar un mecanismo de detección y corrección de errores de propósito general a los enlaces digitales, entendiendo como enlace un único cable que conecta dos máquinas (enlace punto a punto), o varias máquinas (enlace multipunto); este protocolo es muy extenso, por lo que rara vez se utiliza la implementación completa; lo normal es que se utilicen subconjuntos (como el ya mencionado LAP-D).

Como se ve en la figura 10, el HDLC consiste en tramas de bits que están delimitadas por unas banderas de 8 bits de longitud que contienen el valor 01111110 binario. Cuando el receptor encuentra este valor en el canal, comienza la lectura de una trama, lectura que termina cuando vuelve a encontrar este mismo valor. Nótese que una bandera puede indicar, simultáneamente, el final de una trama y el comienzo de la siguiente. Puesto que dentro de la trama, en el campo de datos de usuario puede aparecer este valor, el transmisor insertará automáticamente un bit a 0 detrás de cada bloque de cinco bits a 1; el receptor, a su vez, eliminará cada bit a 0 que siga a un bloque de

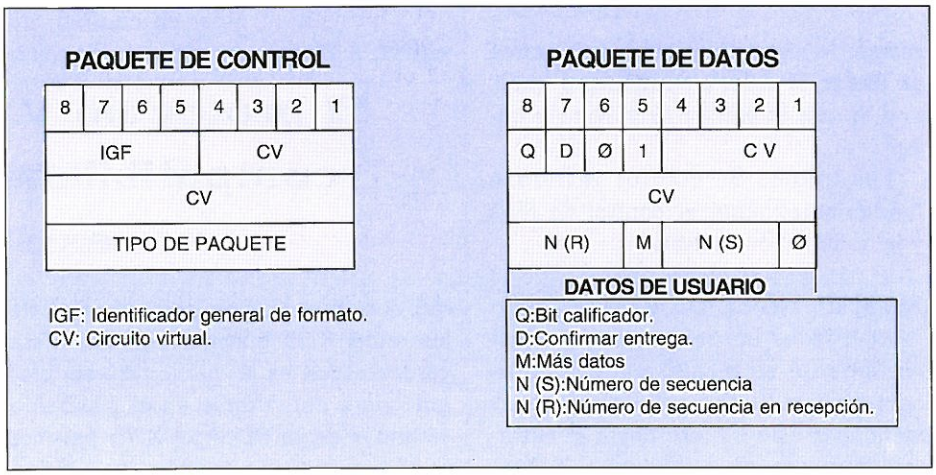


FIGURA 11: Protocolo X.25/X.75.

cinco bits a 1; con este esquema se garantiza que nunca aparecerá el valor de la bandera dentro de los bits de datos; es decir, el usuario puede colocar cualquier información dentro del paquete, la transmisión es totalmente transparente.

El campo de dirección está previsto para sistemas multipunto; en el RDSI se conserva por compatibilidad.

Como se ve en la figura 10, las tramas incorporan una dirección, un código de control y unos números de secuencia. Los números de secuencia de recepción indican el número de secuencia de la siguiente trama que se espera recibir; así, si una trama es recibida correctamente, este valor se incrementará, haciendo que el emisor mande la siguiente trama; si la trama se pierde el valor permanecerá igual, con lo que el emisor la volverá a enviar.

Las tramas de control gestionan fundamentalmente el control de flujo y la notificación de errores.

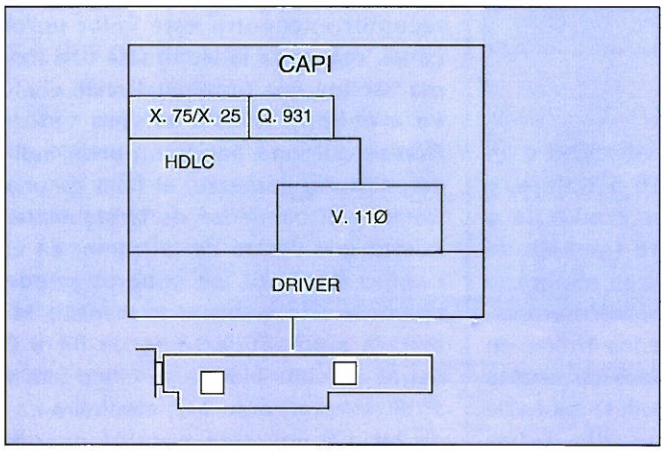


FIGURA 12: Estructura de protocolos RDSI en un ordenador.

X.75, EL NIVEL DE RED

El X.75 es un protocolo que originariamente se diseñó para interconectar redes X.25; funcionalmente, es idéntico a este último, hasta el extremo de que muchos manuales, al describir el X.25, en realidad dan la descripción del X.75.

Este protocolo corresponde al nivel de red, lo que quiere decir que los paquetes pueden pasar por varias máquinas y recorrer varios enlaces antes de llegar a su destino. Es un protocolo orientado a circuitos virtuales (igual que el ATM), cuyos paquetes van encapsulados dentro de paquetes HDLC, por lo que no necesita ningún mecanismo de detección y corrección de errores. El formato de los paquetes X.25/X.75 se puede ver en la figura 11.

Al igual que el HDLC, este protocolo también incluye números de secuencia y confirmación de recepción; en este caso el objetivo es el control de flujo; es decir, garantizar que el emisor no enviará más paquetes de los que el receptor puede procesar.

Aunque el protocolo de paquetes sea idéntico, entre el X.25 y el X.75 existe una diferencia importante; el X.75 soporta múltiples enlaces físicos. El protocolo X.25 se concibió para utilizar un único cable (enlace), el que conecta el ordenador del usuario con la central telefónica; en cambio, un equipo X.75 debe soportar como mínimo dos enlaces, uno por cada una de las redes X.25 que interconecta. Esta característica es la que lo hizo ser elegido para convertirse en el protocolo estándar de la RDSI; el X.75 permite ocultar al usuario la estructura física del RDSI, ofreciendo un mecanismo de canales virtuales que se van mapeando sobre los canales B según sea necesario; por ejemplo, si tenemos un enlace básico (dos canales B) y hacemos dos llamadas al mismo número de teléfono (esto incluye llamadas a diferentes extensiones de un mismo número), ambas llamadas circularán sobre el mismo canal B utilizando diferentes circuitos virtuales; en cambio, si llamamos a diferentes números, entonces se utilizarán los dos canales B.

LA VIDEOCONFERENCIA

El RDSI tiene muchas aplicaciones, cada una con su propio protocolos; está el ya mencionado A-Law para codificación de voz, el G4 que es el protocolo de Fax, o el ya mencionado X.75 para la comunicación entre ordenadores, aunque la aplicación que ha despertado mayor interés es la videoconferencia. La videoconferencia se basa en el protocolo H.320, que define un sistema basado en un tren de imágenes a una resolución de 352x288 a color real, con una velocidad de 30 imágenes por segundo y comprimido utilizando el algoritmo MPEG. Con este esquema se obtiene una calidad de imagen equivalente a la de la televisión, el problema es que el ancho de banda requerido es de (aproximadamente) 2 Mbps (un grupo de 30 canales B). También se ha previsto un sistema de menores requerimientos consistente en reducir la velocidad a 15 imágenes por segundo y la resolución a 176x144, lo que permite reducir el ancho de banda a 384 Kbps (un grupo de 6 canales B). Para poder hacer videoconferencia sobre un enlace básico (a 128Kbps) es necesario, además, suprimir el color; reduciendo aún más las prestaciones, se puede hacer funcionar sobre un único canal B, aunque la calidad resultante es muy inferior al mínimo aceptable.

UTILIZACIÓN DEL RDSI DESDE UN ORDENADOR PERSONAL

Como ya se ha dicho, cuando se

En un enlace básico, la videoconferencia da muy poca calidad de imagen

El protocolo V.110 garantiza la compatibilidad con las redes existentes

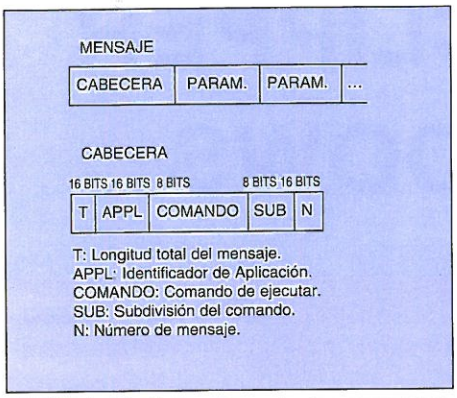


FIGURA 13: Estructura de los paquetes CAPI.

contrata una línea RDSI la compañía telefónica instala un terminal NT-1, siendo responsabilidad del usuario instalar el equipo terminal de conexión en su ordenador. Existen múltiples formas de conectar dicho equipo al ordenador, aunque lo más corriente es que sea una placa que se enchufa en una ranura de expansión.

La mayoría de los terminales RDSI europeos están basados en el juego de chips desarrollado por Siemens, que ofrece el servicio básico; de todas maneras, también pueden encontrarse equipos que ofrecen funciones de valor añadido, como la incorporación de un conector para enchufar teléfonos analógicos convencionales, e incluso la incorporación (en la misma placa) de un modem analógico (un V.32bis por ejemplo); de esta forma, se garantiza la compatibilidad con el sistema telefónico antiguo.

El software necesario para hacer funcionar la placa es un driver para la misma, encima del cual se colocan los protocolos V.110, HDLC, X.75 y Q.931, que se estructuran tal como se muestra en la figura 12. Encima de todos estos protocolos se sitúa la capa mas importante: el CAPI.

En MS-DOS, este software está formado por una serie de programas residentes en memoria, cada uno de los cuales implementa un protocolo.

Este software mencionado SÓLO permitirá utilizar aplicaciones RDSI; para poder utilizar el software existente se necesita un módulo adicional, el CFOSS, que emula un modem Hayes, haciendo posible la utilización de los programas de comunicaciones convencionales sobre RDSI.

CAPI, LA LIBRERÍA DE FUNCIONES ESTÁNDAR

Para manejar dispositivos RDSI desde los programas de aplicaciones se ha creado el CAPI, que es el acrónimo de "Common ISDN API", y define un protocolo que comunica el programa con el driver a través de dos colas de mensajes, una de envío, para los mensajes enviados por la aplicación al driver, y otra de recepción, para los mensajes enviados desde el driver a la aplicación. Si el sistema es multitarea, cada programa que esté en marcha dispondrá de una cola de recepción de mensajes propia; en cambio, la cola de envío es común a todas las aplicaciones. Cuando un programa envía un mensaje, la respuesta a ese mensaje se envía también como un mensaje.

Se ha elegido este sistema porque permite ignorar el mecanismo exacto utilizado para ejecutar las llamadas al driver, lo que lo hace independiente del sistema operativo; de hecho, existen implementaciones de CAPI para MS-DOS, OS/2, UNIX, etc. El CAPI solamente tiene definidas cuatro llamadas a función:

API_REGISTER: Registrar aplicación; está función informa de que una aplicación va a hacer uso del dispositivo RDSI; al ejecutarla se crea una cola de recepción, se realizan las inicializaciones necesarias y devuelve un identificador que deberá ser utilizado en los mensajes.

API_RELEASE: Libera aplicación; está función informa de que la aplicación ya no necesita el dispositivo RDSI; al ejecutarla se liberan los recursos que se hayan podido ocupar y se elimina la cola de recepción.

API_PUT_MESSAGE: Envía un mensaje; esta función coloca un mensaje en la cola de envío.

API_GET_MESSAGE: Recibe un mensaje; esta función lee el siguiente mensaje de la cola de recepción de la aplicación.

FORMATOS DE LOS MENSAJES

Los mensajes se componen de una cabecera seguida de una tabla de parámetros de longitud variable; el formato de los mensajes se puede ver en la figura 13. Cada mensaje debe ir identificado por un número para poder saber a que mensaje se refiere cada mensaje de respuesta.

Para los parámetros se dispone de cuatro tipos básicos de datos, el byte (8 bits), la palabra (16 bits), la doble palabra (32 bits) y la estructura (un byte donde se almacena la longitud en bytes seguido por los datos). El CAPI dispone de mensajes para manejar todos los aspectos de la comunicación RDSI, incluyendo la posibilidad de trabajar sobre canales B directamente, o bien utilizar el protocolo X.75 para trabajar sobre canales virtuales.

RDSI DE BANDA ANCHA. EL FUTURO NO QUIERE ESPERAR

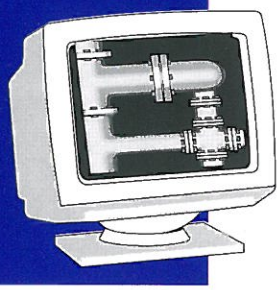
Pese a que la RDSI todavía está a medio construir, los ingenieros ya están pensando en su sucesora: la RDSI de banda ancha. Esta nueva red será, fundamentalmente, idéntica a la RDSI actual con la diferencia de que la velocidad mínima de los canales será de 2 megabits por segundo, pudiendo llegar a los 100. Los investigadores ya están desarrollando toda una nueva gama de aplicaciones para esta nueva tecnología, que podrá sentar las bases de toda una nueva gama de servicios basados en la televisión de alta definición. ■

Puede contactarse con el autor por correo electrónico, en las direcciones:
FidoNet: 2:343/140.22
Internet: monti@bbs-ce.uab.es

Las normas RDSI europeas y americanas son diferentes e incompatibles

FIVEWIN, CLIPPER BAJO WINDOWS

Enrique Velasco



CA-Clipper 5 ha dejado de ser únicamente una plataforma de desarrollo de aplicaciones de gestión bajo entorno MS-DOS, ya que gracias a la librería FIVEWIN ahora es posible desarrollar auténticas aplicaciones Windows con CA-Clipper.

FiveWin permite crear programas Windows en modo nativo sin necesidad de poseer conocimientos exhaustivos de dicho entorno, para ello FiveWin utiliza la potencia de la programación orientada al objeto y el preprocesador de CA-Clipper.

REQUISITOS

Es necesario al menos un ordenador 386 con 2 Mb de RAM para poder utilizar FiveWin tanto por parte del programador como por parte del usuario, aunque un mínimo de 4 Mb es prácticamente obligado.

Para desarrollar programas Windows con FiveWin es necesario usar un linkador que sea capaz de crear ejecutables para Windows, ya que el suministrado por CA-Clipper no es capaz de hacerlo, para ello FiveWin recomienda utilizar el linkador Microsoft LINK (versiones 5.1 o posterior) o Blinker 3.0 (éste último es el más conveniente); el linkador TLINK de Borland no es soportado de momento.

Aunque no es imprescindible también es necesario poseer una herramienta visual de generación de recursos como puede ser el Resource Workshop de Borland con el cual podremos diseñar diálogos, iconos, cursores e imágenes entre otros, para después incluirlos en las aplicaciones que realicemos.

CARACTERÍSTICAS

FiveWin es una librería para CA-Clipper que permite realizar programas bajo Windows, sin embargo dentro del paquete hay mucho más de lo que se pueda ver a simple vista ya que FiveWin proporciona a CA-Clipper un auténtico entorno orientado al objeto con creación de nuevas clases, herencia, polimorfismo, etc.

FiveWin utiliza esta potencia OOP para crear sus propias clases y desde ellas acceder a todo el API de Windows sin que el programador tenga necesidad de entenderse con el mismo, sin embargo y debido a que FiveWin incorpora en el paquete todos los fuentes de dichas clases es realmente sencillo modificar o subclasificar cualquiera de ellas para acomodarla a nuestras necesidades. Además el uso de todas y cada una de las clases se puede realizar utilizando comandos al más puro estilo Dbase, por lo que la utilización de FiveWin es realmente sencilla.

FiveWin le da acceso a casi la totalidad de recursos de Windows como por ejemplo:

- Entorno MDI (Multiple Document Interface)
- Menús flotantes
- Barras de Botones
- Bitmaps (Imágenes)
- Fuentes True Type
- Cursores
- Iconos
- Cajas de diálogo
- Acceso a DLL
- Controles VBX
- Clipboard
- Enlaces DDE
- Ficheros INI
- Metafiles
- Meters
- Brushes
- Sonido (Archivos WAV, MID)
- Vídeo (AVI)
- ODBC

Además FiveWin incorpora la clase TWBrowse para poder realizar browse bajo Windows y la clase TReport para realizar cualquier informe.

INSTALACIÓN

FiveWin sólo ocupa 2 disquetes y para su instalación incorpora un programa el cual crea los directorios necesarios y descomprime los archivos de los que consta FiveWin sin ofrecer apenas ninguna dificultad.

FiveWin incorpora numerosos ejemplos así como gran cantidad de recursos ya creados como iconos, cursores, imágenes, etc. En la instalación se crea un directorio SAMPLES en el cual se incluyen gran cantidad de ejemplos los cuales se convierten en herramienta de consulta imprescindible en los primeros pasos para el desarrollo de aplicaciones Windows. Algunos de los ejemplos no podrá observarlos si no posee el compilador de recursos de Microsoft, el cual permite incluir dentro de nuestros ejecutables recursos generados externamente. El compilador de recursos es una pequeña utilidad de Microsoft (RC.EXE) la cual suele venir incorporada en todos los compiladores de C o C++ para Windows.

EL PRIMER PROGRAMA CON FIVEWIN

Después de observar varios de los ejemplos y en cuestión de cinco minutos ya estamos en condiciones de realizar nuestro primer programa con FiveWin , por ejemplo, el típico programa "Hello World" es tan sencillo como:

```
STATIC oWnd

FUNCTION Main()

DEFINE WINDOW oWnd FROM 0,0 TO 10,10 ;
        TITLE "Mi primer programa con FiveWin"

        @ 1,1 SAY "Hello World"

        ACTIVATE WINDOW oWnd

RETURN NIL
```

Añadir más funcionalidad a nuestro pequeño programa es muy fácil, por ejemplo para indicar que nuestra ventana va a tener un menú sólo hay que crear el objeto menú e incluirlo en la definición de la ventana:

```
DEFINE WINDOW oWnd FROM 0,0 TO 10,10 ;
        TITLE "Mi primer programa con FiveWin" ;
        MENU oMenu
```

Para añadirle además una barra de botones haremos:

```
        DEFINE BUTTONBAR oBar SIZE 30,30 OF oWnd
```

tanto la definición del menú como la funcionalidad de cada uno de los botones de nuestra recién creada barra se hacen con un lenguaje de comandos tan sencillo como el aquí mostrado.

PROGRAMACIÓN POR EVENTOS

Posiblemente haya oído que cuando se empieza a programar bajo Windows cambia completamente la forma de programar, pasando del método clásico o procedural al de programación por eventos, con FiveWin dicho cambio no es tan traumático ya que simplifica mucho el trabajo de programación que debemos hacer, por ejemplo, observemos el siguiente código fuente:

```
ACTIVATE WINDOW oWnd ;
        ON LEFT CLICK LeftClick();
        ON MOVE SeMueve() ;
        VALID MsgYesNo("Desea abandonar la aplicación")
```

Con este sencillo código estamos indicando a una ventana en concreto como debe responder a determinados eventos y sólo para aquellos que queremos controlar ya que para el resto FiveWin se encarga de su gestión. Cada vez que pulsemos el botón izquierdo del ratón encontrándose dentro del área de la ventana se realizará automáticamente una llamada a nuestra función LeftClick(), lo mismo ocurriría para los eventos de movimiento de la ventana (MOVE) y cerrado de la misma (VALID).

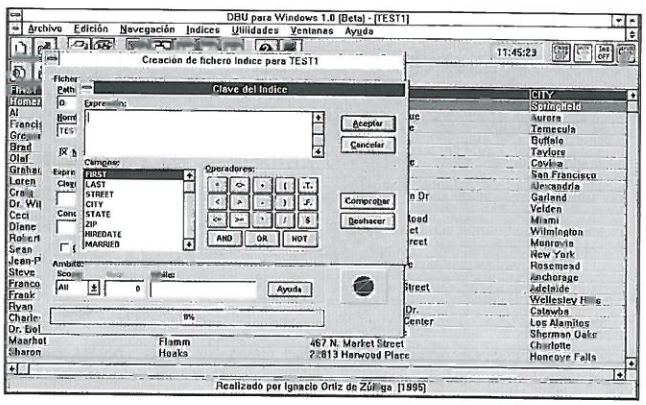
Otra precaución a tener en cuenta es que siempre hay que programar como si nos encontráramos en entornos de red ya que Windows es multitarea, y por tanto es perfectamente factible que un mismo fichero puede estar abierto por varias aplicaciones, o varias veces por la misma aplicación. Si queremos mostrar una base de datos ordenada de dos formas distintas en dos ventanas no tendremos más remedio que abrir dicho fichero dos veces para lo cual debemos de tener la precaución de asignar distintos alias a cada uno de los ficheros abiertos.

Aquellos que utilicen un editor que no sea Windows han de tener en cuenta que la codificación ascii tradicional (OEM) no coincide con la Windows (ANSI) por lo que si desea utilizar su editor de siempre tendrá que utilizar constantemente la función de traslación de OEM a ANSI, por ejemplo:

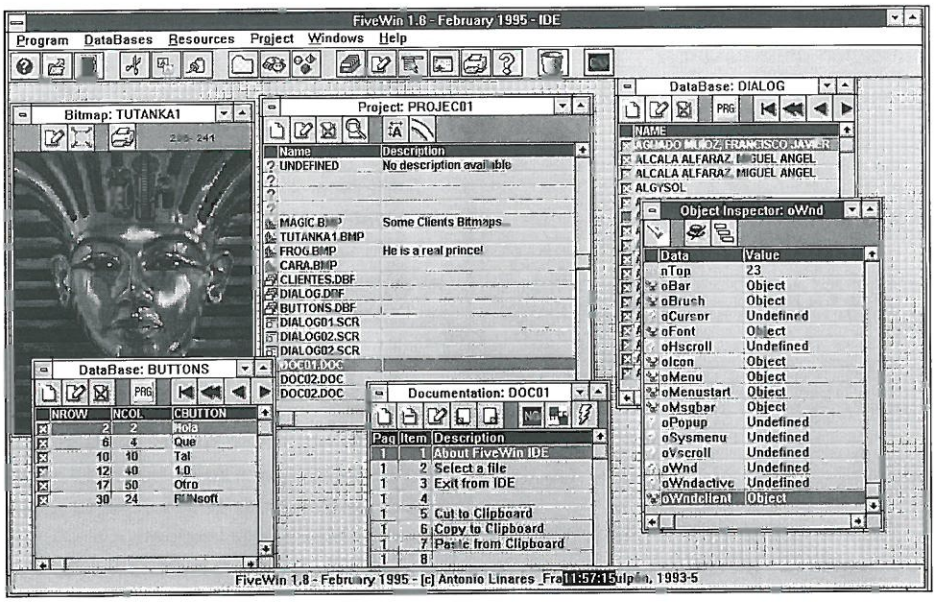
```
VALID MsgYesNo(OemtoAnsi("Desea abandonar la aplicación"))
```

PROGRAMACIÓN ORIENTADA AL OBJETO

CA-Clipper no es realmente un entorno de programación orientado al objeto, ya que adolece del más fundamental de sus requisitos como es la creación de nuevas clases y por lo tanto la de herencia. Sin embargo con el motor OOP que posee FiveWin de nombre OBJECTS obtenemos un entorno de programación completa-



Ejemplo de programa realizado con FiveWin.



Programa IDE incluido con FiveWin.

mente orientado al objeto. OBJECTS no es la única librería que da soporte OOP a CA-Clipper, pero si es una de las más rápidas y potentes, a la última versión de OBJECTS se le han incorporado sobrecarga de operadores al estilo de C++, control de errores definible para cada clases creada, y además sobrecarga de métodos y datos. Con anterioridad ya tenía métodos virtuales, métodos en línea, protección por ámbitos de acceso, tanto para datos como métodos, protección de tipos y variables de clase.

FiveWin viene con cerca 70 clases creadas para acceder a casi toda la potencia de Windows, y es muy posible que gran cantidad de usuarios de FiveWin no hayan creado una clase nunca puesto que sólo utilizan sus objetos que son instancias de dichas clases, y simplemente con ellas obtienen todo lo que necesitan.

DOCUMENTACIÓN

La documentación es uno de los puntos flacos de FiveWin, ya que de momento no existe documentación impresa, y solamente hay una guía electrónica tipo Norton, la cual ha sido mejorada últimamente con la versión 1.7, pero sigue sin ser completa. Sería de agradecer que dicha ayuda estuviera en formato de fichero help de Windows más que en formato Norton. En cualquier caso la falta de documentación se ve compensada por la gran cantidad de ejemplos que se suministran con el producto.

Teniendo en cuenta que FiveWin es un producto español lo lógico es que la documentación también lo fuese y sin embargo no lo es.

Con FiveWin se incluyen la casi totalidad de los fuentes de programación

COMPATIBILIDAD

Cuando hablamos de compatibilidad hay que distinguir entre compatibilidad con FiveWin y compatibilidad con Windows, en principio si una librería es compatible con Windows también es compatible con FiveWin. Para que una librería sea compatible con Windows ha de poder ejecutarse en modo protegido, y además no debe acceder al hardware directamente ni debe utilizar la pantalla ya que es dominio exclusivo de Windows, y por lo tanto a de utilizar los servicios que Windows proporciona para acceder a dichos recursos.

Debido a que FiveWin incorpora su propio motor OOP es incompatible con otros motores OOP existentes como es Class(y) el cual es utilizado por algunas librerías comerciales, por lo que FiveWin es incompatible con Class(y) y con todas aquellas funciones de otras librerías que utilicen un run-time de Class(y), por ejemplo la librería Funcky II lo utiliza, sin embargo se puede usar perfectamente con FiveWin, siempre y cuando no se utilicen las clases que vienen incorporadas en la librería de Funcky II.

Son compatibles con FiveWin las librerías de Comix, Six 2.0 y parte de Funcky entre otras. En cuanto empiece a programar bajo Windows se dará cuenta de que gran cantidad de las librerías que estaba usando ya no son necesarias, porque el propio Windows le da soporte para realizar lo mismo , como por ejemplo librerías gráficas, de acceso a redes locales, envío de faxes, etc. Además descubrirá nuevas formas de librerías como son los controles VBX y las librerías DLL los cuales se pueden utilizar perfectamente desde FiveWin y sin ninguna limitación.

FiveWin convierte a CA-Clipper en un auténtico lenguaje orientado a objetos

SOPORTE TÉCNICO

El soporte técnico se realiza desde Compuserve en el forum de CA-Clipper y a través de la empresa Mail Simmons, bien mediante fax o por su BBS. En la actualidad los tiempos de respuesta son aceptables no excediéndose las 48 horas.

FUENTES

Una de las grandes ventajas de FiveWin es que aporta el 90% de los fuentes tanto los escritos en CA-Clipper como en C, lo cual permite

conocer como esta realmente implementado FiveWin para acceder al API de Windows y así cambiar, modificar y porque no, mejorar cualquier clase o función de FiveWin, es fácil subclasificar cualquier clase de FiveWin para modificar su comportamiento a nuestro antojo, de la misma forma aunque FiveWin no da soporte completo al API de Windows, es realmente sencillo crear una pequeña función en C para acceder a cualquier función del API que no esté implementada.

tabilidad con FiveDos es imposible al 100% por razones obvias, sin embargo hay que modificar muy poco código cuando la conversión se hace desde FiveDos a FiveWin o FiveOS2.

Hay que resaltar que FiveOS2 es hoy por hoy el único producto existente para desarrollar aplicaciones con CA-Clipper bajo dicho entorno.

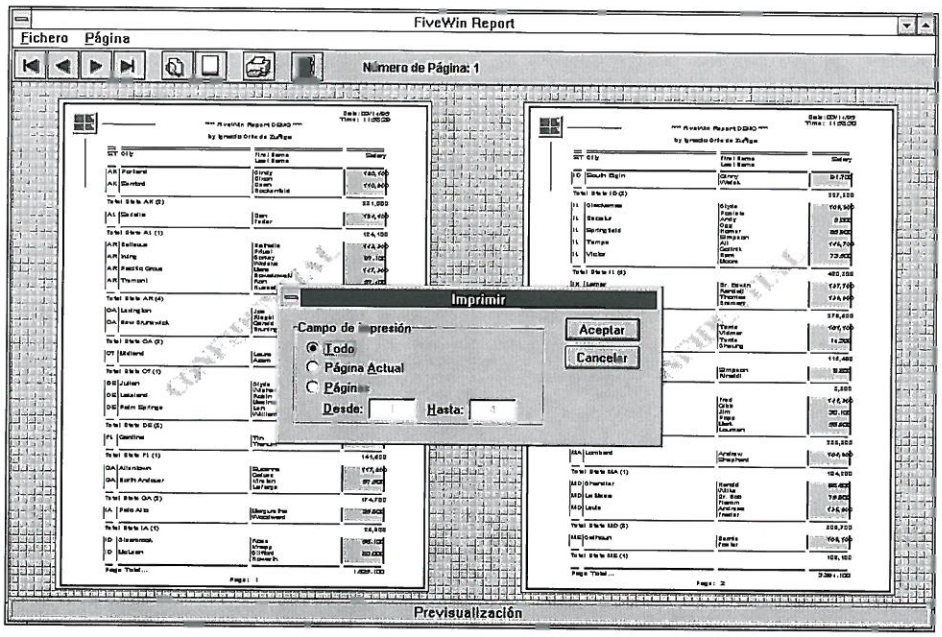
QUE MÁS NECESITA

Si está decidido a trabajar con FiveWin debe tener en cuenta que hay algunas herramientas adicionales que va a necesitar:

- Un linkador para Windows: preferiblemente Blinker 3.0
- Un generador de recursos: preferiblemente Resource WorkShop de Borland
- Un compilador de recursos
- Un compilador de ayudas (si va a realizar ayudas en formato .hlp)

La plataforma FIVE le permite portar sus programas a MSDOS, Windows y OS/2

La incorporación de los fuentes en el producto ha servido para que mucha gente haya mejorado substancialmente partes de FiveWin, que con posterioridad han sido introducidas en las siguientes versiones del producto para beneficio de todos los usuarios.



Presentación preliminar de un listado con FiveWin.

MULTIPLATAFORMA

FiveWin es un miembro de una familia que son: FiveWin, FiveDos y FiveOS2, por lo que sus creadores hablan de la plataforma FIVE, e intentan que exista portabilidad entre aplicaciones DOS, Windows y OS/2. La por-

CONCLUSIONES

Computer Associates acaba de lanzar al mercado lo que se supone va a ser el producto a utilizar si se desea migrar desde CA-Clipper al entorno Windows que es Visual Objects para CA-Clipper, a pesar de que no dudo en absoluto de las grandes posibilidades de dicho producto, he de decir que CA-VO no es la evolución lógica de CA-Clipper que todos esperábamos, ya que se trata de dos productos completamente distintos.

FiveWin ha demostrado que es posible realizar aplicaciones para Windows sin perder la filosofía del lenguaje original de CA-Clipper, utilizando el lenguaje de comandos que tanta fama proporcionó al Dbase, auténtica tecnología OOP y si lo desea sumergiéndose de lleno en el complejo API de Windows.

Hoy por hoy si quiere realizar aplicaciones Windows con el lenguaje CA-Clipper que ya conoce, la solución más acertada es FiveWin. ■

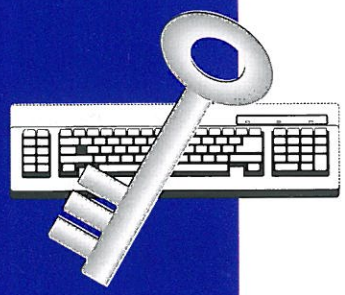
FICHA	
Valoración	90%
Sonido	Si
Animación	Si
Velocidad	Rápida.
Equipo Empleado	486DX-33Mhz, 8 Mb de Ram.
Lo mejor	La facilidad de uso.
Lo peor	Documentación mejorable.
Requerimientos	CPU 386, 2 Mb de RAM, 4 Mb recomendables.
Fabricante	Antonio Linares Software
Distribuidor	Mail Simons, S.L.
Precio PVP	14.000 + I.V.A.

Apartado Correos: 2643, 28080 Madrid
Tf: (91) 563 44 86
Fax: (91) 563 44 51
Bbs: (91) 563 78 72

TÉCNICA

MULTITAREA EN MODO PROTEGIDO

Enrique Velasco



Los sistemas operativos multitarea como OS/DOS o Unix permiten que dentro de un programa se ejecuten varias tareas simultáneamente. APDMX pone esta misma posibilidad al alcance de programas para DOS creados con los compiladores de C/C++ de Borland y Watcom.

Un programador que vaya a desarrollar un juego de acción, u otra aplicación que requiera una alta interactividad y manejo de grandes cantidades de datos, tiene que hacer frente a dos limitaciones fundamentales impuestas por MS-DOS: el techo de los 640 Kb gestionados por el sistema, y la ausencia de servicios de multitarea, que permitan descomponer el programa en varias tareas bien estructuradas.

La manera habitual hasta hace no mucho de aliviar las restricciones de memoria ha sido el empleo de servicios como XMS o EMS. En cuanto a la multitarea, la solución pasaba por la creación de un bucle general que comprobaba variables de estado, e iba llamando una tras otra a las rutinas pertinentes. Esta estructura de "gran bucle", como es bien sabido por todos los que la han empleado, da lugar a programas difíciles de entender, y en consecuencia difíciles también de depurar y de mantener, con el incremento de costes que esta situación conlleva.

La aparición de los *DOS-extenders* de 32 bits ha hecho posible emplear toda la memoria presente en la máquina, tanto para código como para datos, pero en general sin soporte de multitarea. APDMX permite combinar la multitarea con las altas prestaciones del modo protegido, y todo ello sin renunciar a los servicios proporcionados por MS-DOS.

¿QUÉ ES EL MICROKERNEL APDMX?

La parte de un sistema operativo que se encarga de las funciones de planificación y manejo de tareas, suele recibir la denominación de núcleo o *kernel*. Por extensión, se usa también este término, para referirse a las librerías que realizan estas funciones dentro de una aplicación y al margen del sistema operativo.

APDMX está pensado para ser integrado dentro de un programa. MS-DOS y el *DOS-extender* "creen" estar ejecutando una aplicación convencional, aunque ésta internamente esté compuesta de varias tareas concurrentes. El código correspondiente a estas tareas está contenido dentro de funciones C/C++, y al crear una tarea se pasa un puntero a la función correspondiente.

Desde la perspectiva del programador, la parte principal de APDMX se presenta como un fichero OBJ, el cual se enlaza con una aplicación desarrollada con los compiladores de C/C++ de Borland, o de Watcom (ver la ficha del producto). Este fichero contiene el código, que permite la creación y destrucción de tareas, y proporciona los servicios de sincronización y comunicación entre tareas. Existe además un programa residente, que debe ser cargado antes de ejecutar la aplicación, y que realiza el tratamiento de algunas interrupciones hardware.

En la versión registrada se incluye un módulo de depuración que, manteniendo el mismo API realiza comprobaciones sobre parámetros de llamadas y gestión de memoria, así como un perfil de la ejecución de tareas. El módulo de depuración permite además interrumpir en cualquier momento la ejecución para presentar volcados del estado interno de las tareas, los semáforos y la memoria. Estas características son onerosas en términos de consumo de memoria y de tiempo de CPU, por lo que una vez depurado el programa, se sustituye el módulo de depuración por el módulo de producción.

INCORPORACIÓN DE APDMX A UNA APLICACIÓN

En todos los ficheros fuente que hagan llamadas a servicios de APDMX, se debe incluir la cabecera MMKERNEL.H, que contiene los prototipos de las funciones del API. Una vez hecho esto, hay que indicar al *linker* que se desea enlazar el fichero objeto correspondiente. En los entornos integrados, esto es tan sencillo como incluir el nombre del objeto en la lista de ficheros del proyecto. La nomenclatura empleada es la siguiente:

- UKvvrrcd.OBJ : Parte principal de APDMX, es la parte que se enlaza con la aplicación.
- IMvvrrcd.EXE : Programa residente para el tratamiento de interrupciones.

Los dos dígitos *vv* corresponden el número de versión y *rr* al número de *release*. El carácter "c" se refiere al compilador para el que está preparado el fichero: "W" para Watcom y "B" para Borland. Por último, el carácter *d* indica si se trata de un módulo de depuración, o uno de producción, siendo "D" en el primer caso y "P" en el segundo. En la versión 1.1 de APDMX no es necesaria una versión especial de depuración del programa residente, por lo que sólo se entrega la de producción.

DISEÑO

En el diseño de sistemas operativos existen dos enfoques básicos: el de *kernel* monolítico, y el de *microkernel*. El primero consiste en introducir en el *kernel* tantos servicios como sea posible, lo que trae consigo una menor escalabilidad, y mayores dificultades para adaptarlo a máquinas diferentes.

La mayoría de los sistemas operativos modernos, como OS/2, Mach y QNX, están construídos de acuerdo con una filosofía diametralmente opuesta: se dejan en el *kernel* única y exclusivamente los servicios más esenciales, consiguiendo de este modo un *kernel* pequeño y rápido (de ahí el nombre de *microkernel*). El resto de las funciones que llevaría a cabo un *kernel* monolítico quedan a cargo de tareas *del sistema*. Esto permite la adaptación a diferentes configuraciones de hardware sin más que arrancar las tareas del sistema que sean necesarias.

Las mismas dos opciones se presentan al diseñar una librería multitarea. APDMX es un *microkernel*, y el fabricante ha anunciado la aparición de módulos que mediante tareas del sistema implementarán gestión de periféricos y comunicación transparente a través de una red local.

Otra decisión importante es la relativa a la política de planificación a utilizar. Se habla de *preemptive multitasking* (multitarea preemptiva) cuando a una tarea se le puede retirar el control de un modo forzoso, y es característica de sistemas como Unix y OS-2. La política contraria, consistente en que cada tarea ceda explícitamente el control, se denomina *cooperative multitasking* (multitarea cooperativa), y ha sido la adoptada para Windows 3.X, y Novell NetWare. Esta última política de planificación de tareas ha sido también la escogida para APDMX, al considerarse importante hacer mínimo el número de cambios de contexto, lo que aumenta el rendimiento global del sistema.

SERVICIOS QUE PROPORCIONA APDMX

No hay espacio en estas páginas para detallar todas las llamadas del API de APDMX. Se puede encontrar una referencia completa en el manual que acompaña a la versión *shareware* que se ha incluido en el disco de este mes.

Los servicios proporcionados se pueden dividir en los siguientes grupos:

- Manejo de tareas : Estos servicios permiten crear y destruir tareas, así como obtener o cambiar la información asociada a una tarea concreta.
- Sincronización : Las llamadas de este grupo permiten el acceso de varias tareas a un mismo recurso compartido evitando las condiciones de carrera. El principal mecanismo disponible es el semáforo de Dijkstra, pero también se permite la inhibición de interrupciones para secciones críticas de corta duración.
- Temporización : De estos servicios permiten la medida tiempos con una resolución de 5 milisegundos.
- Gestión de memoria: Asignación y liberación de bloques de memoria.
- Comunicación entre tareas : Envío de mensajes entre las diferentes tareas. Este envío de mensajes se puede llevar a cabo con o sin sincronización asociada. Está prevista la ampliación para el envío de mensajes entre tareas en máquinas diferentes a través de una red local.
- Depuración: Estas llamadas sólo funcionan en la versión de depuración, y permiten congelar y reanudar la ejecución de tareas, así como obtener información sobre el estado actual de tareas, semáforos y bloques de memoria. También es posible controlar el funcionamiento del *profiler* de tareas.

En el fichero autoextraíble EXAMPLES.EXE hay varios ficheros con fuentes en C, que hacen uso de los diversos servicios del *kernel*.

DISTRIBUCIÓN Y LICENCIAS

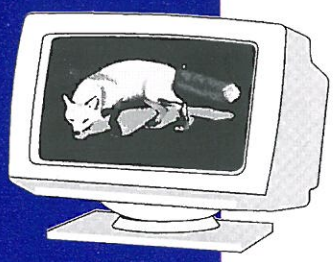
En el disco de este mes se puede encontrar una versión *shareware* de APDMX. Esta versión *shareware* es de libre distribución, y está limitada a 16 tareas y 32 semáforos. Los usuarios que vayan a utilizar APDMX en sus aplicaciones deberán registrarse, con lo cual obtienen la versión completa, que soporta 256 tareas y 256 semáforos. ■

FICHA DE PRODUCTO

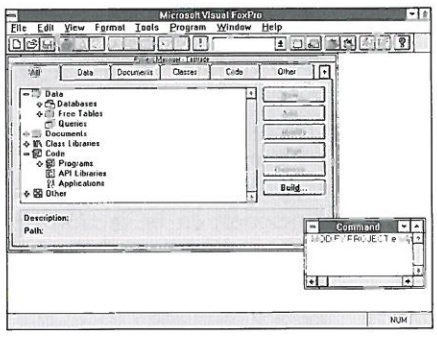
Nombre del producto:
APDMX 1.1
Ordenador:
PC con DOS
Soporte:
Disquetes
Requerimientos:
Ordenador 386 o superior con al menos 4Mb de RAM. Borland C++ 4.02+PowerPack o Watcom C/C++ 10.0 (o versiones posteriores).
Fabricante y distribuidor:
APD S.A.
C/ Antonio López 234 28026 Madrid
Telf: 91-4750412 (Div. Tecnologías de la Información)
Modalidad de distribución: Shareware
Precio de registro:
4000 pts programadores individuales, 9000 pts empresas. Ambos precios incluyen ya el IVA.

VISUAL FOXPRO 3.0 PROFESSIONAL

Fernando de la Villa



Microsoft parece dispuesta a acaparar el mercado con productos de calidad, siempre a la altura de las circunstancias. En este caso se trata de FoxPro, un producto que supo hacerse un hueco entre los desarrolladores de aplicaciones de bases de datos. Esta versión, que ya recibe el nombre de "Visual", incorpora muchas mejoras respecto a la versión 2.6. Microsoft ha apostado fuerte por FoxPro, y atendiendo a las necesidades y exigencias de los desarrolladores ha incorporado algunas de las caracter'sticas más solicitadas por los mismos.



El gestor de proyectos centraliza toda la información relacionada con un proyecto.

Uno de los objetivos principales ha sido el conseguir el Desarrollo Rápido de Aplicaciones (RAD en inglés), que otras compañías como Borland (con Delphi) también persiguen. Se pretende que el desarrollador emplee el menor tiempo posible en crear una aplicación profesional. Se ha dotado a FoxPro de orientación a objetos, se ha mejorado el lenguaje Xbase para dar soporte a esta caracter'stica, y se incluyen diversas herramientas visuales para crear objetos que pueden ser reutilizados. Aparte de ello, ahora se hace un total aprovechamiento del entorno Windows y es posible el desarrollo de aplicaciones dirigidas por eventos.

Otra de las tendencias actuales son los sistemas cliente/servidor. FoxPro proporciona facilidades para la creación de este tipo de aplicaciones de forma local, facilitando más la tarea, para posteriormente trasladar dichas aplicaciones a un entorno real cliente/servidor. También se ha buscado la interoperabilidad con otras aplicaciones, mediante componentes OLE, ampliando las capacidades de FoxPro. Por último, no se han olvidado de los usuarios de versiones anteriores del producto, proporcionando migración automática de las aplicaciones escritas con las versiones 2.x a la nueva versión.

El desarrollo de bases de datos que proporciona FoxPro es de 32 bits, pero puede ejecutarse bajo Windows 16 y 32 bits. Por tanto puede, al igual que sus aplicaciones, correr bajo Windows, Windows para Grupos, Windows NT y podrá ser utilizado con Windows 95.

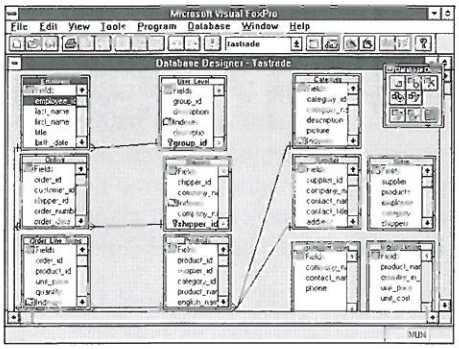
Parecen claras las tendencias de los nuevos productos del mercado informático. El modo texto agoniza, en un momento en el que sólo se habla de entornos gráficos y programación visual. FoxPro es uno de los productos estrella de Microsoft y no podía, ni mucho menos, quedarse atrás en este mundo tan competitivo.

DESARROLLO CON VISUAL FOXPRO

Toda aquella persona que conozca el entorno Visual Basic se verá gratamente sorprendida por el hecho de que el desarrollo de aplicaciones en ambos productos es muy parecido en muchos aspectos. Microsoft parece buscar una forma de manejo similar en todos sus programas, de tal modo que el tiempo empleado en el aprendizaje de un nuevo producto sea el más pequeño posible. La idea no es mala, sobre todo si se piensa en la cantidad de dinero que anualmente invierten las empresas en la formación de sus empleados.

Con esta nueva versión, los usuarios pueden aprovechar todas las características que ofrece el entorno Windows a los programas. Si se sigue el modelo de las aplicaciones conducidas por eventos, es el propio sistema el que se encarga de manejar muchos de los eventos.

El desarrollo en Visual FoxPro se ha mejorado, soportando el paradigma de la programación orientada a objetos. Se permite la creación de clases y objetos



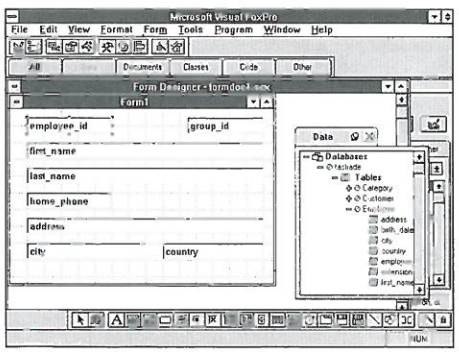
En el diseñador de bases de datos se ven en pantalla las tablas, vistas y relaciones existentes.

de forma visual, y no visual (mediante sentencias). Existe un diseñador visual de clases que permite la creación de clases y subclases sin necesidad de conocer la sintaxis al respecto. Las características de programación orientada a objetos incluidas son: herencia, polimorfismo y encapsulación. A pesar de todo ello, se sigue dando soporte a la programación procedural de Xbase.

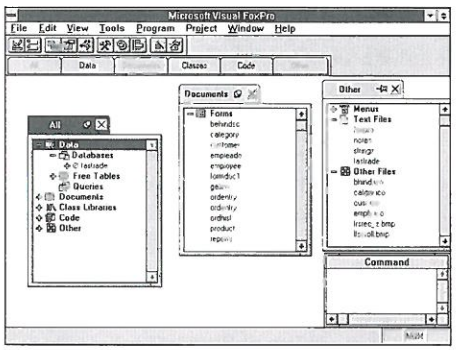
Pero el peso de gran parte del trabajo "sucio" en la creación de una aplicación de bases de datos recae sobre las herramientas visuales que acompañan al programa. Entre ellas figuran el gestor de proyectos, el diseñador de bases de datos, el de formularios y el diseñador visual de clases.

EL GESTOR DE PROYECTOS

Este es el corazón del desarrollo de una aplicación bajo Visual FoxPro 3.0. El gestor de proyectos centraliza toda la información y todos los ficheros relacionados con un proyecto. Lo que contiene se encuentra distribuido en seis separadores: todo, datos, documentos, clases, código y otros. La información en cada separador está organizada en forma jerárquica, de forma similar a como estructuran los directorios muchos gestores de ficheros. Se dispone de una raíz de la que cuelgan los elementos principales. Cada elemento puede ser ampliado o reducido con la pulsación del signo más o el signo menos situado junto al nombre de los elementos que contienen información. Así, en forma de árbol, se puede alcanzar el nivel de detalle deseado. La ventana del gestor de proyectos se puede reducir a una barra formada por las solapas de los separadores, situándola junto a las barras de herramientas. Con esto se proporciona acceso inmediato a cualquier elemento de la aplicación cuando se necesita. Para hacer más cómodo el acceso, los separadores se pueden ejecutar desde el gestor, arrastrando dicho separador a cualquier parte de la pantalla. Otra característica de los separadores es la chincheta, presentada una vez arrancados, que posibilita la permanencia del separador por encima de las demás ventanas en todo momento.



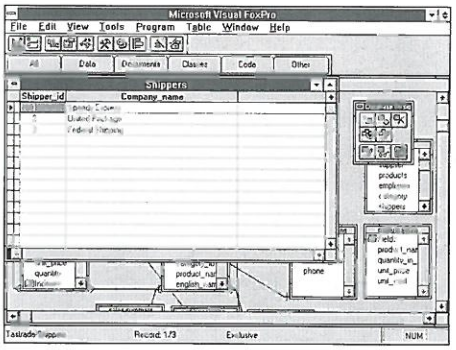
El diseñador de formularios ayuda al diseño del aspecto de la aplicación.



En esta pantalla se pueden apreciar algunos de los separadores ejecutables del gestor de proyectos.

A los usuarios de OS/2 les resultará familiar lo siguiente: si se pulsa con el botón derecho del ratón cualquier objeto presentado en un separador, aparecerá un menú con opciones. Entre las acciones presentadas aparecen algunas como: Nuevo, Añadir, Modificar, Borrar, etc. Esta nueva funcionalidad permite esconder menús que sólo aparecen cuando se necesitan, evitando una sobrecarga innecesaria de la pantalla.

Pero la gran potencia del gestor de proyectos y sus separadores, es la posibilidad de arrastrar y soltar (drag and drop) los objetos que contienen hacia los formularios e informes.



Desde el diseñador de bases de datos también se puede ver el contenido de las tablas.

EL DISEÑADOR DE BASES DE DATOS

Otra de las herramientas visuales de FoxPro es el diseñador de bases de datos. Como su nombre indica, permite diseñar cómodamente las tablas, vistas y relaciones que forman la base de datos, siempre de forma visual. Por ejemplo, para crear una relación entre dos tablas basta con señalar uno de los índices de una de las tablas, y arrastrarlo hacia el campo clave adecuado en la otra tabla.

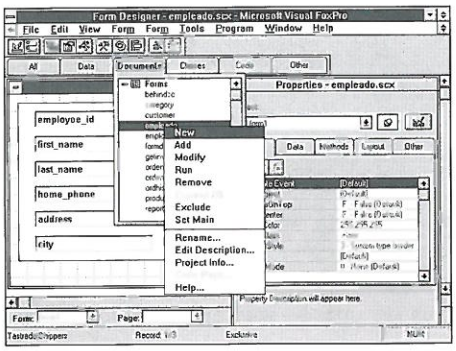
Automáticamente aparecerá una ventana, el constructor de relaciones, con opciones que posibilitan la especificación de las características que definen a esa relación. Las relaciones establecidas en el diseñador de bases de datos son persistentes durante toda la aplicación, permitiendo el establecimiento automático de relaciones entre tablas a la hora de crear formularios e informes.

EL DISEÑADOR DE FORMULARIOS

El aspecto de una aplicación, y su interactuación con el usuario depende casi de forma exclusiva de los formularios. El diseñador de formularios es muy similar al de Visual Basic.Los objetos se arrastran, colocan y redimensionan de igual modo, y cada objeto posee una lista de propiedades, presentada en ventana aparte, que modifican el comportamiento y aspecto de los mismos. La característica más destacada de este diseñador es la posibilidad de crear una clase con un grupo de objetos situados en el formulario, clase que puede ser usada posteriormente en otros formularios.

EL DISEÑADOR VISUAL DE CLASES

Este diseñador es muy similar al anterior en cuanto a apariencia se refiere. Entre otras cosas, ofrece al usuario la posibilidad de crear propiedades y métodos propios. FoxPro incorpora en esta versión nuevos controles, más potentes que los de la versión 2.6. Algunos de ellos, como la rejilla (Grid Control) puede ser grabado después de su creación para ser utilizado en nuevos formularios. Otro de los nuevos controles es la estructura de página (Page Frame Control), utilizado para presentar la información de las tablas en forma de páginas.



La creación de relaciones entre tablas se realiza de forma visual, con gran simplicidad.

DESARROLLO CLIENTE/SERVIDOR

Para facilitar el desarrollo de aplicaciones cliente/servidor, FoxPro proporciona herramientas de conectividad. Los drivers ODBC que incorpora dan acceso a bases de datos de diferentes formatos como Oracle, dBase, Access, Excel, Microsoft SQL Server y Btrieve, entre otros. Admite la ejecución de procesos por lotes, además de poder definir vistas sobre tablas remotas de forma local, para coger del servidor sólo la información necesaria. Dispone también de un diseñador de conexiones en donde se especifica la manera de acceder a una fuente de datos determinada.

INTEROPERABILIDAD

FoxPro dispone de un soporte completo de OLE, y de los interfaces TAPI y MAPI. Esta característica le permite interactuar con objetos OLE como si fueran propios de FoxPro. Los controles OLE, también soportados, pueden ser integrados en subclases como si se tratara de cualquier otro objeto.

OTRAS CARACTERÍSTICAS

Además de lo ya descrito, Visual FoxPro ofrece otra serie de características. Como novedad, ahora dispone de un diccionario de datos, para permitir una mayor integridad en los datos. Sobre las tablas se pueden definir una serie de reglas que son aplicadas de forma automática en cualquier lugar en que éstas sean utilizadas. Junto con el diccionario, el contenedor de bases de datos y el entorno de datos permiten tener un mayor control sobre la información manejada por la aplicación y posibilitan la realización automática de algunas operaciones relacionadas con formularios e informes.

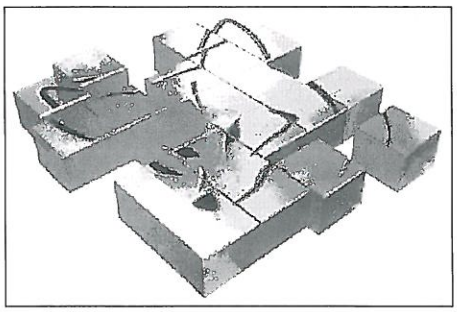
Para ayudar al desarrollador en muchas de las tareas inherentes a la creación de una aplicación, FoxPro 3.0 incluye multitud de wizards o utilidades. Entre ellos se pueden destacar:

- Form Wizard: Sirve para crear formularios a partir de una tabla.
- View Wizard: Crea una vista con datos locales.
- Remote View Wizard: Crea una vista con datos remotos.
- Query Wizard: Para crear un acceso selectivo a los datos
- Label Wizard: Se usa para la creación de etiquetas de correo
- Graph Wizard: Crea un gráfico en Microsoft Graph que muestra información sobre los datos.
- Group/Total Wizard: Genera un informe resumen que proporciona totales para grupos de registros.
- PivotTable Wizard: Envía datos desde FoxPro hacia una tabla pivote de Microsoft Excel y puede crear un formulario con la tabla.
- Upsizing Wizard: Crea una base de datos Oracle o Microsoft SQL Server a partir de una base de datos FoxPro.
- Report Wizard: Se utiliza para generar informes a partir de una tabla.
- Documenting Wizard: Analiza y reformatea código fuente.
- Import Wizard: Permite traspasar datos de otras aplicaciones al formato de FoxPro.
- Setup Wizard: Genera un programa de instalación para las aplicaciones.
- Table Wizard: Crea una nueva tabla con los campos especificados.

Otras herramientas disponibles son los constructores. Se utilizan para establecer propiedades o construir expresiones en el momento del diseño de la aplicación. Se presentan en forma de ventana con diversas opciones disponibles, que varían según el tipo del constructor.

EN DEFINITIVA

Uno de los aspectos positivos de FoxPro es la comodidad de trabajo que proporciona, permitiendo al usuario organizar el escritorio de la forma que más le convenga. Su manejo resulta intuitivo y simple. El acceso a la información se produce de manera muy rápida gracias al uso de la tecnología Rushmore de acceso a datos, por lo que el manejo de grandes cantidades de datos no resulta un gran problema para FoxPro. El mayor problema, como comienza a ser habitual en los tiempos que corren, es la utilización de mucha memoria. De hecho, se recomienda un mínimo de 8 Mb de RAM en el ordenador en el que se piensa utilizar. Como puede apreciarse, FoxPro constituye una completa herramienta para el desarrollo de aplicaciones de bases de datos bajo Windows. La inclusión, entre otras cosas, de la programación orientada a objetos, OLE y ODBC le convierte en un producto actual y competitivo. ■



FICHA DEL PRODUCTO

Nombre: **Visual FoxPro 3.0 Profesional**
Fabricante: **Microsoft**
Requerimientos mínimos:
- Windows 3.1, **Windows para Grupos, Windows NT o Windows 95**
- Procesador: **386 o superior**
- Ratón
- 8 Mb de RAM
- 11 Mb de disco duro
- Se recomienda VGA

Precio: Aún no suministrado por Microsoft.

USO DE LA MEMORIA EXTENDIDA

Fernando Pérez

Una de las limitaciones más grandes de la programación en modo real es la memoria. ¿ Por qué programando bajo C++ solo se puede usar 640 Kb, si se posee un ordenador con 4 Mb?. En este artículo se intentará "poner las cartas sobre la mesa", y desvelar pequeños secretos de la memoria extendida y su uso.

El MS-DOS facilita el manejo de la memoria extendida a través de su controlador HIMEM.SYS. Éste se encarga de reservar, mover y liberar memoria por nosotros, así como, de ofrecer información sobre el estado de la memoria. Para operar con este controlador se necesita unos mínimos de información, saber si esta cargado en memoria y qué dirección física ocupa, de esto se encarga la interrupción compartida 2FH. Una simple llamada a esta interrupción con el registro AX cargado con el valor 4300H informará de la presencia o ausencia del controlador.

UN POCO DE TÉCNICA

La creación de una clase en C++ para el soporte de memoria extendida no es muy complicado, sabiendo cuándo, dónde y qué se debe utilizar. En el libro "Extensiones del DOS" se relatan todas las funciones que soporta el controlador de XMS, y qué valores se han de entregar. Sólo hay que tener en cuenta la imposibilidad de direccionar una variable de clase, teniendo que usar una variable intermedia.

¿ DÓNDE ESTÁ EL CONTROLADOR?

Ante todo, es fundamental comprobar la existencia del controlador para evitar posibles "cuelgues". Anteriormente se citó la forma de llevar esto a cabo mediante la interrupción 2FH. Veamos a continuación un ejemplo:

```
unsigned char nPresente;
    _asm {
        MOV  AX,0x04300;
        INT  0x02F;
        MOV  nPresente,AL;
    }
```

La interrupción devolverá en AL el valor 0x080 si el controlador está cargado u otro valor si no lo está. Este valor es pasado a la variable *nPresente* para evaluar su contenido en C++.

Inmediatamente, si el controlador está en memoria, se obtendrá la dirección física que ocupa este. Una vez mas se recurrirá a la interrupción 2FH, pero esta vez se llamará al servicio 43H, subservicio 10H, devolviendo la posición en los registros ES:BX. Ésta es la porción de código:

```
unsigned long XMSDIR;

    _asm{
        PUSH ES;
        MOV  AX,0x04310;
        INT  0x02F;
        MOV  WORD PTR XMSDIR,BX;
        MOV  WORD PTR XMSDIR+2,ES;
        POP  ES;
    }
```

Una vez recogida la dirección del controlador, se debe guardar en una variable C para su posterior uso, y ésta es *XMSDIR*. La forma de cargar la variable es un poco peculiar, pero efectiva.

LLAMANDO AL CONTROLADOR XMS

Conociendo la posición física del controlador XMS, llamar a sus funciones, es tan fácil como llamar a una interrupción, se carga en AH la función a realizar y se llama al controlador, ésta sería la forma correcta.

En este ejemplo, se llama al controlador para ver la cantidad de memoria libre de la que se dispone.

```
unsigned TOTXMSFREE;
            _asm{
        MOV  AX,0x08;
        CALL DWORD PTR XMSDIR;
        MOV  TOTXMSFREE,DX;
    }
```

El total de memoria extendida libre de que se dispone se cargaría en la variable *TOTXMSFREE*.

LA CLASE XMSSUPP

XmsSupp es la encargada de: comprobar si existe el controlador, obtener la dirección de éste, averiguar su versión, memoria extendida libre y realizar un informe visual, sólo heredando la clase. El usuario sólo debe reservar la memoria extendida, la zona de intercambio de la memoria convencional, entregar el puntero de esta zona a la clase, y realizar operaciones con la memoria. Sobre todo, no se debe olvidar liberar la memoria, claro que siempre se puede crear un destructor para los olvidadizos.

La estructura de la clase se puede ver en la figura 1, así como la definición del tipo de estructura necesaria para realizar operaciones de movimiento de memoria.

En primer lugar se puede observar la definición de la estructura *XMSBlock*, utilizada para realizar los movimientos de memoria. Seguidamente comienza la implementación de la clase. *ThisXMSBlock* es la estructura del tipo *XMSBlock*, que usará cada herencia de la clase; *lError* es una variable de tipo lógico, que indica si se ha producido un error en alguna operación, por eso es conveniente comprobar el estado de ésta, después de cada llamada a un método de la clase; *cError* es una cadena de caracteres, que informa acerca del mensaje de error si se produce. Las demás variables son de uso interno de la clase, y si algún lector está interesado en su uso, sólo debe consultar el código de la clase.

La estructura *XMSBlock*, no debe ser utilizada, a no ser que se tenga muy claro lo que quiere hacerse, ya que la clase se encarga de actualizar los valores en todo momento.

No todos los métodos de la clase deben ser utilizados por el programador, realmente sólo son necesarios *AllocXms()*, *SetConPntr()*, *Move2Xms()*, *Move2Con()* y *FreeXms()*. *AllocXms()* se encarga de reservar memoria XMS, y se recibe como parámetro el número de Kb a reservar. Por ejemplo, si queremos reservar 64 Kb se usará:

```
XmsSupp MiXmsSupp;
MiXmsSupp.AllocXms(64);
```

Es aconsejable comprobar el estado de *lError* para observar cualquier anomalía en la ejecución del método. *SetConPntr()* se encarga de actualizar el puntero *SrcHandle* de la estructura *XMSBlock*, que apunta a una zona de memoria previamente reservada, donde se realizaran las acciones de carga y descarga de la memoria XMS.

Una de las limitaciones más grandes de la programación en modo real es la memoria

```
unsigned char far *ZonaCDC;
ZonaCDC=new unsigned char far [64*1024];

MiXmsSupp.SetConPntr((void far *)ZonaCDC);
```

Move2Xms() mueve el bloque de memoria convencional a memoria extendida, y *Move2Con()* realiza la acción inversa.

```
MiXmsSupp.Move2Xms();
...
...
MiXmsSupp.Move2Con();
```

FreeXms() es la encargada de liberar la memoria anteriormente reservada. Como antes se citaba, se puede crear un destructor de clase que llame a la función.

```
MiXmsSupp.FreeXms();
```

UN EJEMPLO

Para demostrar el buen funcionamiento de la clase, se tomó como ejemplo un programa en C++, qué mueve una pantalla a XMS, y la vuelca a la VGA.

En primer lugar se debe reservar los 64 Kb de memoria extendida que se usará para almacenar la imagen, además de la memoria convencional para la creación de la misma. Una vez almacenada la imagen en XMS, se cambiará el puntero de origen direccionándolo a la memoria de VGA, y se realiza la operación de movimiento a convencional, *Move2Con()*. Se cambia el puntero de origen ya que la clase se encarga de intercambiar los *handles*, y el desplazamiento de origen y destino, según la operación de movimiento a realizar.

Sólo queda una cosa por reseñar, los fuentes están totalmente comentados y es muy fácil comprender el funcionamiento de la clase en todo momento. ■

FICHERO DE CABECERA XMSSUPP.H

```
#ifndef XMSSUPP_H
#define XMSSUPP_H

// Definición de bloque para el manejo de la memoria XMS

typedef struct _XMSBlock{
        unsigned long   Len;
        unsigned        SrcHandle;
        unsigned long   SrcDesp;
        unsigned        DstHandle;
        unsigned long   DstDesp;
}XMSBlock;

// clase para el manejo de la memoria XMS

class XmsSupp
{
public:
        XMSBlock        ThisXMSBlock;
        unsigned        lError;
        char            cError[80];
        unsigned        Present;
        unsigned        Reserved;
        unsigned long   XmsDir;
        char            HiVerXms;
        char            LoVerXms;
        char            HiVerCtr;
        char            LoVerCtr;
        unsigned        IHMA;
        unsigned        BigBlock;
        unsigned        TotXmsFree;

                        XmsSupp();
        void            InitXms();
        void            GetInfo();
        void            GetXmsDir();
        void            GetXmsVer();
        void            AllocXms(unsigned);
        void            GetXmsFree();
        void            SetConPntr(void far *);
        void            FreeXms();
        void            Move2Xms();
        void            Move2Con();
        void            GetXmsErr(char);
};
#endif
```

Figura 1: Listado del fichero XMSSUPP.H

TERMINALES REMOTOS EN LINUX

David Aparicio

Unix es un sistema que permite conectar varios usuarios simultáneamente en la misma máquina. Para ello, es obvio que se necesitan pantallas y teclados independientes para cada usuario. Un PC estándar permite, sin ningún hardware especial, tener hasta tres usuarios simultáneos: Uno en la pantalla principal, o consola, y dos más conectados mediante los puertos serie, mediante terminales.

EL HARDWARE

Para conectar directamente un terminal por línea serie, es necesario conseguir un cable, denominado módem nulo. Básicamente, un cable de estas características cruza las líneas de control y datos de transmisión de un lado con las correspondientes de recepción en el otro, haciendo creer a cada punto que el otro es un módem.

Las señales de un conector DB25 (de 25 contactos) son:

Pin 2 TD (salida):
Transmisión de datos
Pin 3 RD (entrada):
Recepción de datos
Pin 7 GND (común):
Nivel de referencia, o neutro.

Con los tres contactos anteriores, ya es posible realizar una conexión. Durante la comunicación, cuando un lado está recibiendo datos mas rápido de lo que puede procesar, indica al otro que pare mediante un carácter de control (XOFF). Una vez que esta listo de nuevo, lo comunica mediante otro carácter (XON). El software de comunicaciones se encarga de distinguir los propios datos de las secuencias de control. Por todo lo anterior, a este protocolo se le denomina XON/XOFF, o de control de flujo software. Este método tiene el problema de que, si el carácter XOFF no se envía a tiempo, pueden perderse datos, y éstos deben retransmitirse. Esto provoca una pérdida de rendimiento en la conexión.

Para evitarlo, hay señales adicionales:

Pin 4 RTS (salida):
Petición de envío al módem.
Pin 5 CTS (entrada):
Módem listo para enviar.
Pin 6 DSR (entrada):
Módem preparado.
Pin 20 DTR (salida):
Ordenador preparado.

Mediante las señales anteriores, los dos lados pueden sincronizar el enlace de forma fiable. Cuando se quiere solicitar una pausa al emisor, se le indica variando las señales 5 y/o 6. Este método hardware, o RTS/CTS, es mucho más fiable, y es preferible siempre que se disponga de un cable adecuado.

Por último, otras señales que se usan para indicar el establecimiento de la conexión:

Pin 8 CD (entrada) Detección de conexión remota.

El software suele utilizar la señal CD para saber cuándo iniciar el diálogo, es decir, cuándo hay un interlocutor disponible al otro lado del enlace.

La forma de combinar todas las señales anteriores para obtener un cable adecuado se muestran en la figura 1.

EL SOFTWARE

Una vez que se dispone del cable, se debe configurar un ordenador para

En este apartado, se explicará el modo de conectar un terminal a un PC ejecutando Linux.

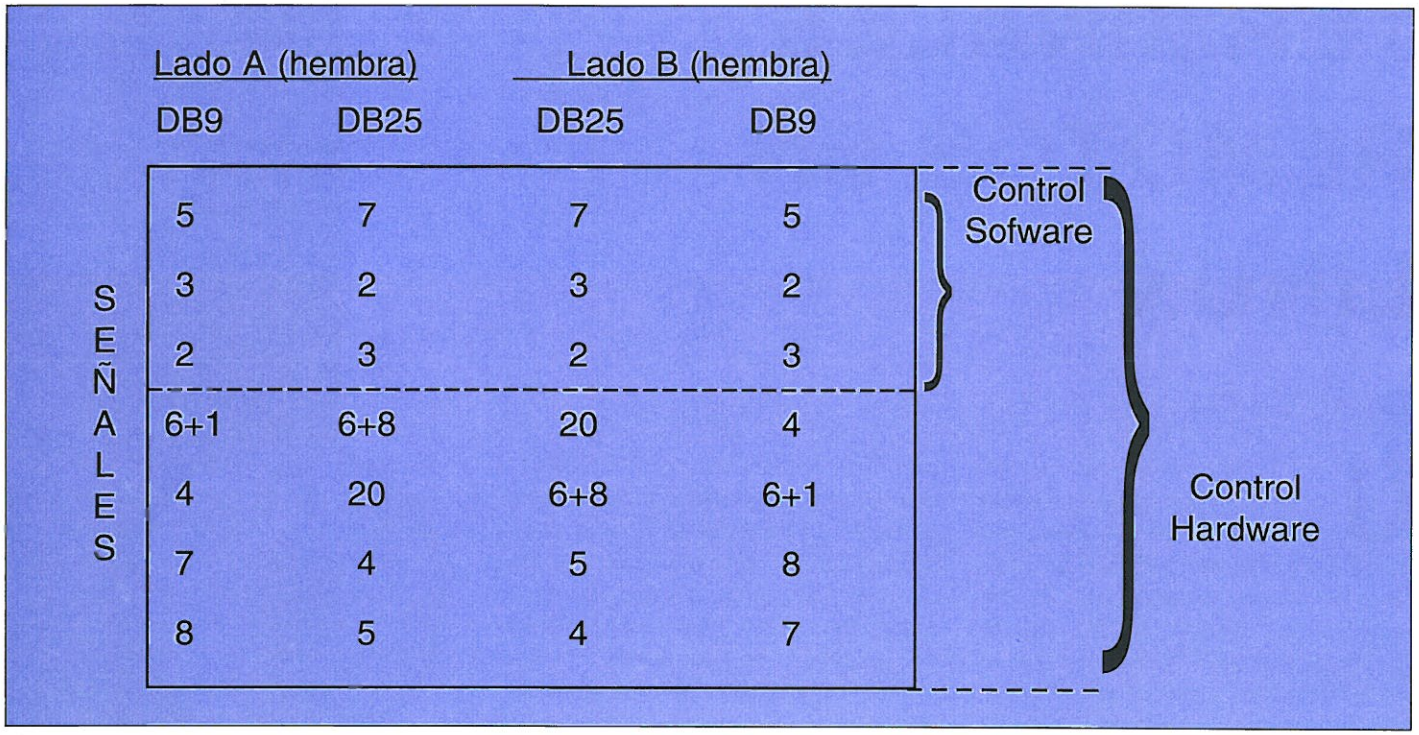


Figura 1: Cable de módem nulo.

esperar la conexión y ofrecer la pantalla de login.

Este lado, denominado pasivo (servidor), será la máquina en la que ejecuta Linux. El programa Unix denominado, genéricamente, *getty*, se encarga de asignar un terminal lógico a un dispositivo físico de entrada/salida. Cuando getty detecta la señal CD, sabe que hay un terminal al otro lado, con lo que muestra la pantalla de entrada. Podemos ver este proceso en la figura 2.

Por otra parte, se necesita un lado activo (cliente), que inicie el diálogo, y permita introducir comandos y obtener resultados.

Para ello, se debe disponer de un programa de comunicaciones, que sea capaz de emular algún terminal reconocido por el servidor. Casi todos disponen de la emulación más común, la denominada VT100.

En la máquina Unix se debe modificar el fichero /etc/inittab, buscando las referencias a los terminales de línea serie, e introduciendo unas líneas de la siguiente forma:

```
...
# Serial lines
s1:45:respawn:/sbin/agetty -hL 19200 ttyS0
s2:45:respawn:/sbin/agetty -hL 19200 ttyS1
...
```

1) El primer campo indica una etiqueta que debe identificar de forma única a cada línea.

Es aleatoria, pero no debe repetirse a lo largo del fichero.

2) El segundo, "45" indica, que esta línea está disponible en los niveles 4 y 5. El comando *init* permite alterar el nivel de ejecución de Linux, controlando así el número de terminales disponibles en cualquier momento.

3) *respawn* significa que getty debe invocarse cada vez que quede libre la línea. Una vez que el terminal cierra la sesión, esta opción indica que se debe lanzar un nuevo getty que atienda al puerto serie.

4) El último campo especifica el programa que controlará la línea junto con sus parámetros.

El programa concreto que se va a emplear es *agetty*, el estándar en Linux. Sus ventajas son:

- Es fácil de configurar
- Ocupa poca memoria.
- Es ideal para líneas directas (sin módem)

La opción "h" habilita el control de flujo hardware. La opción "L" indica que se ignore la señal CD, muy útil cuando nuestro cable no dispone de dicho contacto.

Tras modificar el fichero y ejecutar el comando init q, podemos comprobar que agetty está funcionando mediante el comando ps -x | grep ttyS1. El sistema debe responder con una línea similar a:

```
#ps-x | grep ttyS1
103 pS1 S  0:00 /sbin/agetty -hL 19200 ttyS1
```

Una vez realizado esto, basta con conectar ambas máquinas mediante el cable. Ajustando los parámetros adecuados en el programa de comunicaciones, como velocidad (19200) y protocolo de contención (hardware), debemos poder obtener la pantalla con el login.

Al entrar, se puede comprobar el tipo de terminal que se está emulando, mediante echo $TERM. En Linux, el sistema responderá con "vt100" o "console".

Llegados a este punto, debemos indicar que Linux dispone de dos dispositivos lógicos para cada línea serie. Para el puerto 'X' tendremos /dev/cuaX y /dev/ttySX. El primer dis-

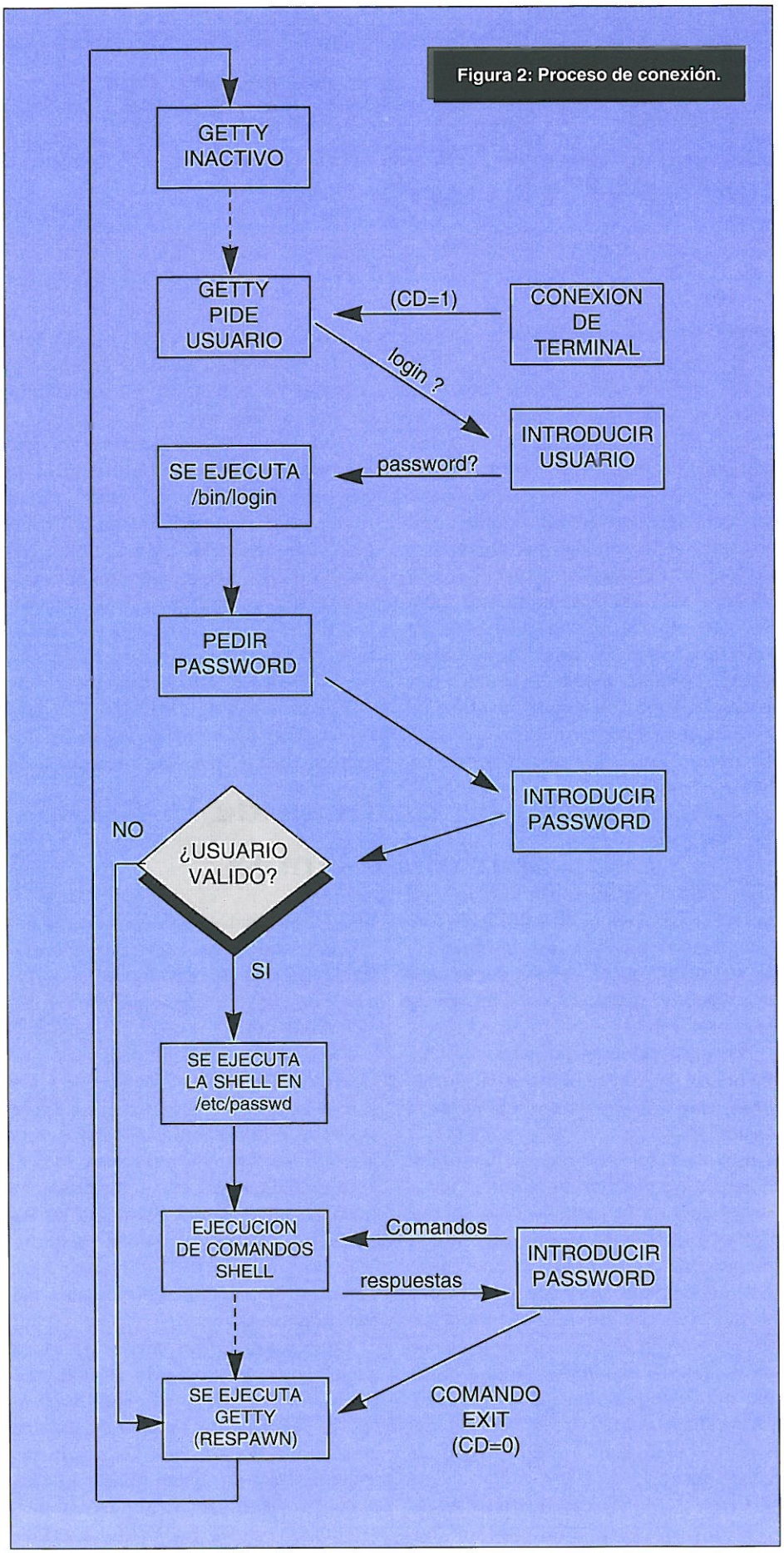


Figura 2: Proceso de conexión.

positivo no tiene en cuenta las señales de control de la línea, y es útil para el manejo como cliente. El segundo se utiliza cuando nuestro sistema actúa como servidor, dado que se requiere conocer el estado de CD para saber cuándo se está solicitando una conexión.

Esta dualidad tiene mayor sentido cuando se emplea un módem, donde se requiere su uso para enganchar, como terminal, con otro sistema sin entrar en conflicto con el getty que pueda estar permanentemente esperando atender llamadas entrantes.

Si se conecta como terminal otro sistema Linux, se puede usar *minicom* (un clónico del popular Telix) como emulador. Algunos sistemas Unix también disponen de *kermit,* otro clásico programa de comunicaciones.

Por último, para transferir ficheros, se pueden emplear los comandos sz y rz para enviar y recibir, respectivamente, desde el nodo servidor, con el protocolo zmodem. Todo esto se encuentra en /linux210/a3/comms.tgz.

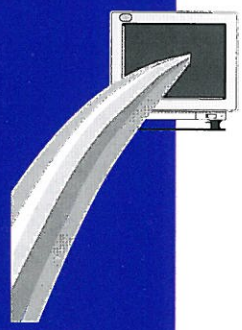
CONCLUSIONES

El método explicado aquí permite comprender el funcionamiento básico del acceso remoto en Unix. Como siempre, se recomienda consultar la documentación adicional, y la documentación incluida en man sobre el particular.

Una vez que se maneje con soltura esta configuración, se puede probar a mejorarla mediante un programa más sofisticado, que se encuentra en el CD, en /linux210/a2/getty.tgz.

Para conectar un módem, en lugar de un cable directo, sólo se deben variar las opciones con las que se invoca getty. Revise el fichero /etc/inittab, el cual contiene líneas comentadas destinadas a tal efecto. Por último, recuerde añadir el comando ATS0=1 a la configuración de su módem, para que éste pueda descolgar el teléfono.

Mediante estas sencillas operaciones, el lector podrá comprobar lo fácil que resulta empezar a sacar partido a las capacidades multiusuario que Unix puede ofrecerle. ■



LOS VGA MODOS GRÁFICOS (I)

Sergio Ríos

En esta ocasión nos ocuparemos de los modos gráficos de la VGA, haciendo especial hincapié en el modo 13h, el más sencillo de los modos de 256 colores.

Tal como vimos en la anterior entrega de este curso, en los modos alfanuméricos la unidad de información que se maneja es el carácter. Es el hardware de la VGA el encargado de traducir el código del carácter en el conjunto de puntos (matriz del carácter) mediante los cuales se representa dicho carácter por pantalla.

En los modos alfanuméricos, se considera a este conjunto de puntos como un bloque o una unidad indisoluble. Por ejemplo, todos los pixels que componen un carácter, tienen las mismas propiedades en lo que a color se refiere.

Además, recordemos que en estos modos no es posible situar los caracteres en posiciones arbitrarias de la pantalla, sino que existen restricciones para su ubicación (en función del tamaño del bloque o matriz de carácter que se esté usando).

En los modos gráficos, por el contrario, es posible controlar de forma individual el color de cada pixel de la pantalla.

Quiere esto decir que en los modos gráficos se pueden direccionar todos los puntos de la pantalla (eso sí, este direccionamiento puede ser más o menos complejo, en función del modo gráfico considerado). De esta forma, desaparece cualquier limitación o restricción en lo que a la manipulación de la pantalla se refiere: ahora es posible dibujar cualquier figura geométrica, realizar animaciones, representar superficies con gradientes de color...etc.

Lamentablemente, la libertad y el control que se ganan al usar los modos gráficos suponen un incremento de la complejidad de los programas y, sobre todo, una importante disminución de la velocidad.

Para poner de manifiesto estos problemas, es necesario considerar la cantidad de datos que se utilizan para representar una pantalla de información tanto en modo alfanumérico como en modo gráfico. Por ejemplo, para mostrar una pantalla de 80 columnas y 25 filas de caracteres de 16 colores en modo alfanumérico se necesitan 80 * 25 * 2 = 4000 bytes.

En un modo de vídeo gráfico, para representar un texto de idénticas características se usa una resolución de 640 * 480 puntos, y cada punto se representa mediante 4 bits (16 colores). Haciendo cálculos, podemos comprobar que en este caso se necesitan 128000 bytes.

Las funciones gráficas de la BIOS son MUY lentas

Resulta obvio que un programa que manipule 128000 puntos deberá ser forzosamente más complejo y lento que otro que únicamente deba manejar 4000 bytes. De cualquier forma, esta penalización en el rendimiento queda compensada con el uso de los potentes microprocesadores disponibles hoy día (80486, Pentium), que operan a frecuencias de reloj cada vez más elevadas.

En el caso de los modos gráficos supervga, este problema es aún más grave, puesto que la cantidad de puntos se incrementa notablemente con respecto a los modos VGA estándar. En este caso, se suelen utilizar ayudas más específicas, como son los modernos procesadores gráficos disponibles

en las tarjetas aceleradoras de vídeo (S3, Mach, TIGA...)

REPRESENTACIÓN Y DIRECCIONAMIENTO DE PIXELS

Como ya hemos visto, cada pixel tiene asociado un número de bits, que determina el número máximo de colores diferentes que se pueden usar en un determinado modo gráfico. El valor de los bits correspondientes a un pixel indica, de forma directa o indirecta, el color que se utiliza para representarlo en pantalla.

TABLA

Modo de Vídeo	Resolución	Bits por pixel	Colores
4, 5	320 x 200	2	4
6	640 x 200	1	2
0Dh	320 x 200	4	16
0Eh	640 x 200	4	16
0Fh	640 x 350	2	4
10h	640 x 350	4	16
11h	640 x 480	1	2
12h	640 x 480	4	16
13h	320 x 200	8	256

En la tabla 1 se muestran los modos gráficos de la VGA, y se indican las resoluciones usadas, y el número de bits por pixel. Como podemos comprobar, en una VGA estándar nunca son necesarios más de 8 bits para representar un pixel. Quiere esto decir que, como máximo, en determinados modos gráficos de la VGA podremos representar hasta 256 colores diferentes.

Antes comentamos que en los modos gráficos es posible direccionar pixels individuales. Lo que sucede es que existen varios tipos de direccionamiento, y cada modo gráfico utiliza un determinado método.

En cualquier caso, en la memoria de vídeo se almacenan los valores de los bits correspondientes a los pixels de izquierda a derecha y de arriba a abajo en direcciones crecientes. Por tanto, la dirección más baja es la referida al pixel situado en la esquina superior izquierda de la pantalla. Veamos cuáles son los tipos de direccionamiento de pixels:

El modo 13h es el único modo estándar de 256 colores en la VGA

A) Formato empaquetado

En los modos gráficos que usan este método de direccionamiento, los pixels se representan usando 1, 2, o 4 bits (son modos de 2, 4, 16 colores, respectivamente).

En este caso, un byte de la memoria de vídeo almacena la información de 2, 4 u 8 pixels contiguos, donde los bits más significativos corresponden al pixel situado más a la izquierda.

B) Planos de bits

Los modos gráficos nativos de la VGA almacenan los valores de los pixels de forma radicalmente distinta a la anterior. En este caso, cada byte de la memoria de vídeo almacena el valor de uno de los bits de 8 pixels contiguos. Se usan tantos planos como bits por pixel se hayan definido en un determinado modo de vídeo.

El resultado es que tenemos varios planos de bits, que representan una componente (valor de uno de los bits) de la imagen completa. Esta representación puede parecer bastante extraña, pero, como veremos en la siguiente entrega de este curso, permite la manipulación de los pixels mediante técnicas muy eficientes, asistidas por el hardware.
La utilización de varios planos de bits tiene una gran ventaja: soluciona de forma sencilla el problema que supone el direccionamiento de la memoria de vídeo en un espacio limitado de direcciones de la CPU. Por ejemplo, en todos los modos gráficos que hacen uso de planos de bits, es posible acceder a 256 Kb de memoria de vídeo desde un segmento de 64 Kb de la memoria superior (A0000h-AFFFFh)

C) Formatos de 256 colores

En este caso, cada byte contiene directamente la información relativa a un pixel.

Realmente, éste es un caso híbrido, puesto que conjuga algunas características de los dos anteriores; En efecto, es un formato "empaquetado" por cuanto se almacenan los 8 bits de un pixel en un byte. Por otra parte, los modos gráficos de 256 colores usan los cuatro planos de bits disponibles en la VGA.

Sólo existe un modo estándar de 256 colores, el modo 13h. Este modo es sumamente sencillo, ya que el uso de los planos de bits es totalmente transparente al programador.

Existen otros modos gráficos de 256 colores no reconocidos oficialmente bajo ningún estándar, que se consiguen mediante una cuidadosa reprogramación del hardware de la VGA. Mención especial merece el que se conoce como "Modo X", que se ha convertido "de facto" en un estándar de 256 colores en la industria del videojuego por sus inmejorables características para la realización de

CUADRO 1

Código ensamblador para la activación del modo 13h

```
MOV AH,0        ; Selección de modo
MOV AL, 13h     ; Modo 13h
INT 10h ; Servicios de la BIOS
```

CUADRO 2

Función 0Ch / INT 10h: dibuja un pixel en una determinada posición y con el color especificado:

Registros:

AH	0Ch
AL	color
CX	posición horizontal
DX	posición vertical
BH	número de página (en modos con más de una página)

animaciones. Las posibilidades de este modo tan especial justifican por sí solas un capítulo completo de este curso.

EL MODO 13h

Este modo posee una notable característica: su sencillez, debida fundamentalmente al sistema de direccionamiento lineal que ofrece.

En efecto, como se puede observar en la figura 1, la VGA ubica los bytes correspondientes a pixels contiguos en planos distintos. Sin embargo, los pixels contiguos tienen asignadas externamente direcciones de memoria consecutivas.

Este modo gráfico tiene una resolución de 320*200 puntos, lo que equivale a 64000 bytes por pantalla. La memoria de vídeo que usa este modo empieza en la dirección A000:0000h y corresponde al punto (0,0). El último punto (319,199) tiene asignada la dirección A0000:F9FFh (ver figura 2).

La BIOS ofrece una interfaz muy sencilla para el acceso a funciones gráficas, a través de la interrupción 10h. Sin embargo, las rutinas que se utilizan son sumamente lentas.

Por lo general, si se desea obtener el máximo rendimiento a la hora de implementar rutinas gráficas, es prácticamente obligatorio manipular directamente la memoria de vídeo.

De todas formas, existen algunos servicios de la INT 10h que, por su complejidad conviene usar. Es el caso de la función de establecimiento de modos de vídeo.

En el cuadro 1 se puede comprobar lo sencillo que resulta activar el modo 13h usando esta función (AH=0) desde el lenguaje ensamblador. No hay que olvidar que siempre se debe volver al modo texto (AL=3) al finalizar nuestros programas.

CALCULO DE DIRECCIONES

Llegados a este punto, ya tenemos activado el modo 13h. Ahora necesitamos saber cómo se puede dibujar un punto. Para ello, tenemos dos opciones: La primera consiste en usar el servicio 0Ch de la INT 10h (ver cuadro 2), y la segunda, modificar directamente la memoria de vídeo. Como cabe esperar, nos decantamos por la segunda opción, para lo cual necesitaremos saber cuál es, dentro de la memoria de vídeo, la dirección del byte correspondiente a un determinado pixel.

En este modo, la organización lineal de la memoria de vídeo nos permite

CUADRO 3

Código en C:

```
void far *dir_mem;
void far *dir_base;

activar_modo13h();
dir_base = 0xA0000;
dir_mem = dir_base + ( 320 * pos_y ) + pos_x;

*(dir_mem) = color;
```

Código ASM:

```
MOV AX,A000h
MOV ES,AX
MOV AX,( 320 * POS_Y ) + POS_X
MOV DI,AX
MOV [ES:DI],color
```

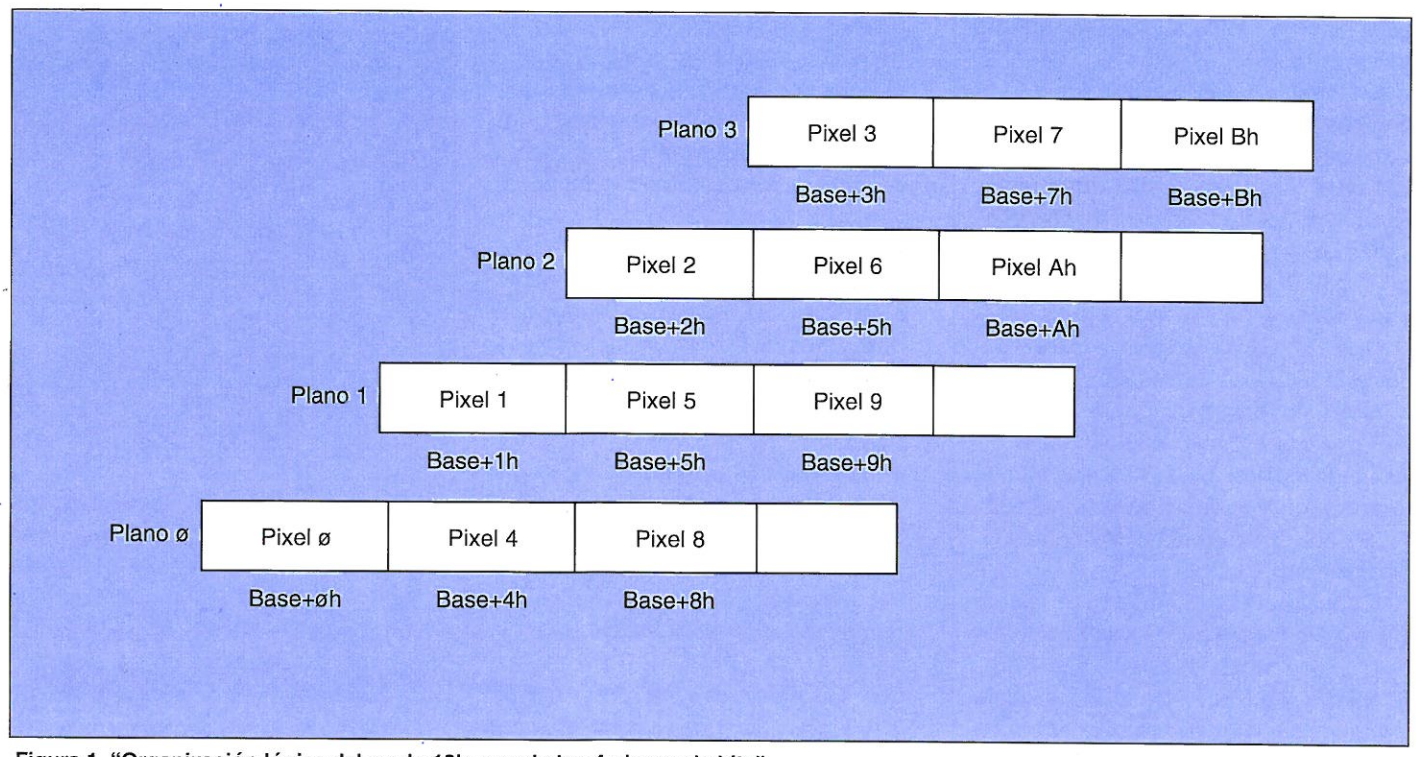


Figura 1. "Organización lógica del modo 13h usando los 4 planos de bits".

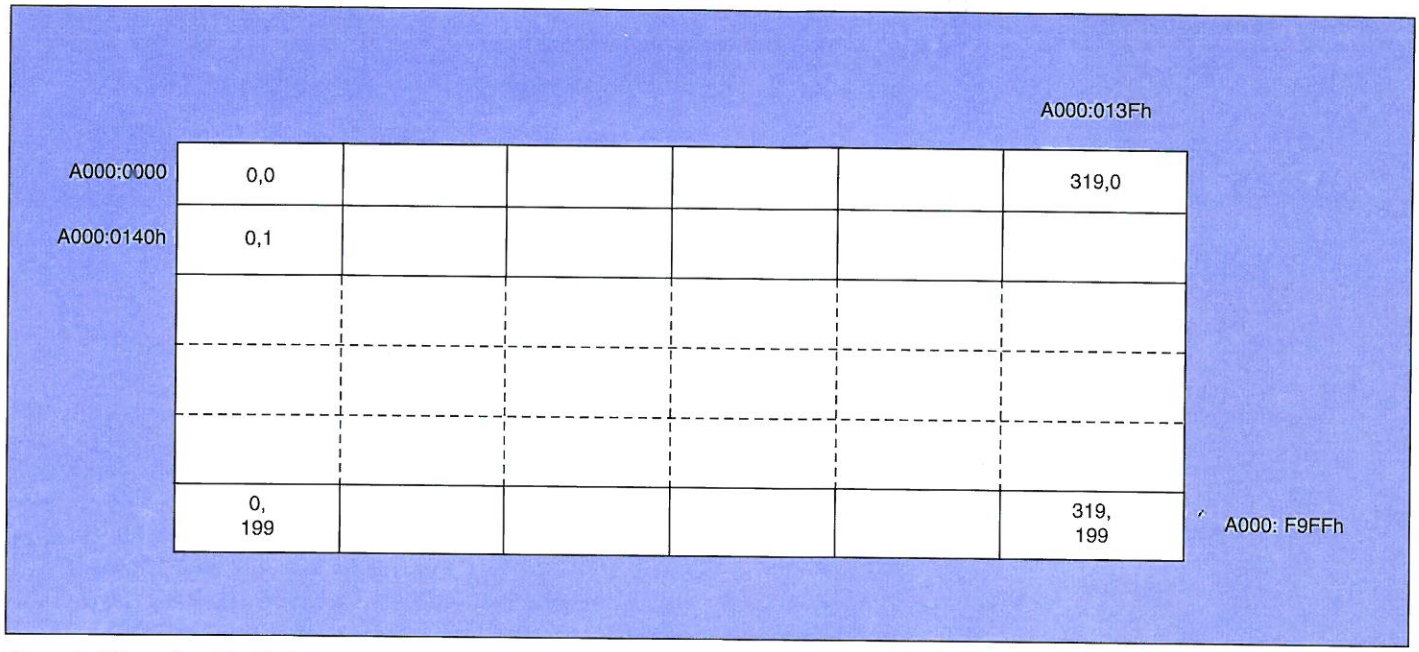


Figura 2. "Organización lógica del modo 13h (tal como resulta visible al programador)"

calcular de forma sencilla la dirección buscada:

Dir_Memoria= A0000h + (320 * coord_y) + coord_x

En esta dirección deberemos guardar un byte cuyo valor indica el color del pixel a dibujar. En unos instantes nos ocuparemos del significado de este valor.

En el cuadro 3 se muestran sendos fragmentos de código en ensamblador y C que permiten dibujar un pixel accediendo directamente a la memoria de vídeo.

Se puede observar que en la fórmula antes indicada para el cálculo de la dirección de memoria asociada a un pixel es necesario realizar una multiplicación, que es una operación costosa en términos de computación.

Podemos intentar reelaborar dicha fórmula a fin de obtener una expresión más conveniente:

Dir_Memoria= A0000h + (256 * coord_y) + (64 * coord_y) + coord_x

Esta nueva expresión de la fórmula nos permite optimizar notablemente el código para dibujar pixels. En efecto, hemos desglosado la multiplicación original en una suma de productos. Lo importante es que estas nuevas multiplicaciones se pueden implementar como operaciones de desplazamiento de 8 y 6 bits, respectivamente, cuyo tiempo de ejecución es considerablemente menor que el de una multiplicación. En el cuadro 4 se muestran los fragmentos de código optimizado.

El modo 13h configura una paleta de colores por defecto

CUADRO 4

Código ASM optimizado

```
MOV AX,A000h
MOV ES,AX
MOV AH, POS_Y
XOR AL,AL        ; AX= POS_Y * 256
MOV BX, POS_X
ADD BX,AX        ; BX= (POS_Y*256)+POS_X
SHR AX,2         ; AX= POS_Y * 64
ADD AX,BX        ; AX= (POS_Y*320) + POS_X
MOV DI,AX
MOV [ES:DI],color
```

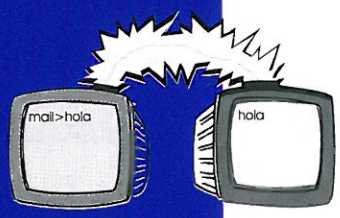
COLORES

Es el momento de abordar un asunto pendiente: los valores que puede tomar un pixel. Cada uno de los 256 valores posibles que puede tener un pixel se corresponde con uno de los registros de color del conversor digital-analógico (DAC). Los registros de color del DAC contienen 18 bits que se utilizan para componer la señal de vídeo analógica que se envía al monitor. Estos 18 bits se desglosan en tres grupos de 6 bits para cada una de las componentes RGB del color. El conjunto de los 256 registros de color recibe el nombre de paleta de colores.

Cuando se activa el modo de vídeo 13h, la BIOS programa los registros de color del DAC con unos valores predeterminados, y cuya organización es la que se muestra en la figura 2. Los registros 0 a 0Fh contienen colores compatibles con la gama de la CGA, por razones de compatibilidad. Los 16 registros siguientes (10h a 1Fh) contienen una escala de grises. Los 216 colores siguientes se dividen en tres grupos de 72 colores, donde los grupos difieren en la intensidad de los colores. Dentro de cada grupo aparecen las componentes básicas (RGB) con una saturación decreciente. En una próxima entrega de este curso nos ocuparemos específicamente de la gestión de la paleta de colores de la VGA, lo que nos permitirá implementar algunos efectos especiales. ■

NETBIOS (y 4ª PARTE)

Carlos Arias



Como en capítulos anteriores, para asentar las ideas que se han ido dando acerca del interfaz NetBios, se va a desarrollar un ejemplo práctico. La aplicación consistirá en una utilidad para que dos ordenadores puedan dialogar entre sí mediante la utilización de datagramas.

Se utilizarán los dos tipos de comunicaciones vistos a lo largo de los capítulos anteriores, y que son: comunicación síncrona y asíncrona. De esta manera el lector podrá apreciar la diferencia de operación entre ambos tipos. Conforme se vaya avanzando en la explicación del ejemplo, se irán indicando las razones por las que se escoge en un determinado momento un tipo de comunicación u otra.

CONCEPTOS PRELIMINARES

Los pasos que sigue el programa para comenzar, mantener y cerrar la conexión entre los ordenadores son los siguientes:

Comprobación de la existencia de NetBios.

Obtención del nombre del ordenador.

Añadir un nombre a la tabla de nombres.

Búsqueda de un usuario.

Preparar al ordenador para recibir datagramas.

Envío mediante datagramas de los mensajes introducidos por el teclado.

Recepción de datagramas.

Cierre de la comunicación y Borrado del nombre.

La comunicación con el controlador de NetBios se realiza a través de la interrupción 5Ch y los registros BX y ES (ver figura 1). Los buffers que se utilizan en la aplicación tienen un tamaño de 255 bytes. El límite de caracteres por cada mensaje a enviar es de 255. Para el envío y recepción de datagramas se han utilizado 2 bloques NCB llamados ncbSend y ncbReceive.

COMPROBAR LA EXISTENCIA DE NETBIOS

De las operaciones anteriores, las dos primeras son fundamentales. La comprobación de la existencia de NetBios se realiza mediante la interrupción indocumentada 2Ah, tal y como se muestra en la figura 2. Hay que recordar que es la única manera de asegurarse que el valor devuelto por la llamada a la función DOS 5E00h no pertenece a un controlador desinstalado.

La comprobación de la existencia de NetBios se realiza mediante la interrupción indocumentada 2Ah

A continuación se obtiene el nombre del ordenador local mediante la llamada a la función 5E00h. Si ya tiene un nombre asignado, se da opción para utilizarlo en la aplicación, evitando el paso de tener que dar de alta un nuevo nombre en la tabla de nombres. Mediante la utilización de la variable de estado flagName, se indica si al finalizar la aplicación hay que borrar el nombre de la tabla, o por el contrario hay que respetarlo. Si se indicó anteriormente que se utilice el nombre dado al ordenador para enviar y recibir datagramas durante la aplicación, y luego se intenta borrar dicho nombre, las consecuencias pueden ser desastrosas, pudiendo llevar desde el blo-

```
/***
        Realiza una llamada al controlador de
NETBIOS
        Parámetros de entrada: Dirección de la
estructura NCB.
        Salida:    contenido del reg. AX.  0=
correcto.
***/
int NbCall(NCB far * ncb)
{
        struct SREGS sregs;
        union REGS regs;
        regs.x.bx = FP_OFF(ncb);
        sregs.es = FP_SEG(ncb);
        return  (int86x(0x5C,&regs,  &regs,
&sregs));
}
```

Figura 1. Llamada al controlador NetBios.

queo del sistema, hasta la completa pérdida de control.

DAR DE ALTA UN NOMBRE EN LA TABLA DE NOMBRES

El proceso que se sigue es el siguiente: Pide un nombre a dar de alta y

```
/* Detecta presencia de NETBIOS mediante inte-
rrupción
  indocumentada 0x2A */

        regs.x.ax = 0;
        int86(0x2A, &regs, &regs);
        if (!regs.x.ax)
        {       printf ("NETBIOS No está
instalado");
                return 1;
        }
```

Figura 2. Detección de NetBios.

llama a la función correspondiente. Si el nombre ya existe, hay dos caminos, o volver a introducir otro nombre, o finalizar la aplicación de manera inmediata. Tanto para ejecutar este comando como en la obtención del nombre del ordenador, se ha utilizado la comunicación síncrona, ya que no hace falta que la aplicación aproveche el tiempo para realizar otras operaciones. La limitación que tiene este modo de comunicación, es que parece que el ordenador se ha quedado "bloqueado", mientras espera que NetBios devuelva el control a la aplicación, lo que puede resultar un tanto incómodo. Una variante del programa para que funcione

El comando NetBios se ejecuta en modo asíncrono, permitiendo interrumpir la búsqueda mediante la pulsación de una tecla

en modo asíncrono y pueda representar una figura en movimiento en pantalla, se recoge en la figura 4. Como regla general, siempre que se pueda, se utilizarán comandos asíncronos en vez de síncronos. De esta manera, además de poder representar algún tipo de movimiento en la pantalla, se puede ofrecer al usuario la posibilidad de cancelar la operación mediante la pulsación de una tecla.

BÚSQUEDA DE UN USUARIO

Mediante la búsqueda del ordenador remoto se indica el usuario con el que se desea establecer el diálogo. Los parámetros que se le pasan a la función son el número de tarjeta adaptadora que se va a utilizar y el nombre del ordenador remoto. El comando NetBios se ejecuta en modo asíncrono, permitiendo interrumpir la búsqueda mediante la pulsación de una tecla. El ordenador se queda en un bucle de espera examinando el valor del campo ncb.retCode. En el momento que el valor del campo es 0 indica que el comando ha sido ejecutado. Hay que hacer una pun-

```
/* Obtiene nombre del ordenador local */
regs.x.ax = 0x5E00;
regs.x.dx = FP_OFF(ncbSend.nameLocal);
sregs.ds = FP_SEG(ncbSend.nameLocal);
intdosx(&regs, &regs, &sregs);
if (regs.x.cflag)
{       printf ("NETBIOS No está instalado");
        return 1;
}
ncbSend.nameLocal[strlen(ncbSend.nameLocal)]=
32;
if (regs.h.ch)
{
        printf ("\nNombre de la estación local:
%s\n", ncbSend.nameLocal);
        puts ("¿Desea utilizar el nombre de la
estación para la aplicación ?:");
        if( (tecla=getche())!='S' && tecla!='s')
                if (NbAddName(&ncbSend))
                        return 1;
}
else
        if (NbAddName(&ncbSend))
                return 1;
```

Figura 3. Obtención del nombre del ordenador.

tualización respecto a este comando: el emulador NetBios de la red Local peer-to-peer Novell Lite 1.1 no trae soporte para esta función, devolviendo un código de error 3, que significa comando NetBios no válido (al final del artículo se incluye una lista de códigos devueltos por el controlador NetBios después de enviarle una orden).

```
/* Da de alta el nombre en la tabla de nombres */
while (1)
{
memset (Name, 32, 16);
 fflush (stdin);
scanf ("%s", Name);
Name[strlen(Name)]=32;
ncb->command=0x30 | 0x80;  /* Se pone a 1 el octavo
 bit para indicar modo asíncrono*/
ncb->lanNumber=0;
_fmemcpy (ncb->nameLocal, Name, 16);
NbCall (ncb);
while (ncb->retCode!=0)
FuncDibuja(); /* Dibuja un objeto mientras se completa el comando*/
}
```

Figura 4. Variante asíncrona para dar de alta un nombre.

PREPARACIÓN DE RECEPCIÓN DE DATAGRAMAS

Para el envío y recepción de información en la aplicación se ha optado por la utilización del datagrama al ser el método más sencillo de intercambio de mensajes. Como se recordará de capítulos anteriores, la utilización de datagramas es una comunicación no orientada a conexión, por tanto, no hace falta la apertura de ningún circuito virtual. El único requisito necesario es que el ordenador que quiera recibir un datagrama ejecute el comando NetBios NbReceiveDatagram.

```
void _loadds interrupt far ESR (unsigned es, unsigned ds,unsigned  di, unsigned  si, unsigned  bp,
 unsigned  sp, unsigned bx,unsigned dx, unsigned cx, unsigned  ax,
 unsigned  ip, unsigned cs, unsigned  flags)
{
cuerpo de la función...
}
```

Figura 5.

En el programa se ha especificado además en el NCB la dirección de la rutina ESR para que se ejecute de forma automática cuando se reciban datagramas. Hay que realizar las siguientes consideraciones a la hora de realizar operaciones dentro de esta función:

Puesto que se desactivan las interrupciones mientras se está en la rutina, conviene que las operaciones se realicen lo más rápidamente posible.

Si se intenta borrar de la tabla de nombres el nombre dado al ordenador, las consecuencias pueden ser desastrosas

Si se van a realizar muchas operaciones con el stack conviene definir uno propio y restaurar a la salida el original.

```
00h successful
01h bad buffer size
03h invalid NETBIOS command
05h timeout
06h receive buffer too small
07h No-ACK command failed
08h bad session number
09h LAN card out of memory
0Ah session closed
0Bh command has been cancelled
0Dh name already exists
0Eh local name table full
0Fh name still in use, can't delete
11h local session table full
12h remote PC not listening
13h bad NCB_NUM field
14h no answer to CALL or no such remote
15h name not in local name table
16h duplicate name
17h bad delete
18h abnormal end
19h name error, multiple identical names in use
1Ah bad packet
21h network card busy
22h too many commands queued
23h bad LAN card number
24h command finished while cancelling
26h command can't be cancelled
30h name defined by another process (OS/2)
 34h NetBIOS environment not defined, must issue reset (OS/2)
 35h required operating system resources exhausted (OS/2)
36h maximum applications exceeded (OS/2)
37h no SAPs available for NetBIOS (OS/2)
38h requested resources not available (OS/2)
40h Lana System Error
41h Lana Remote Hot Carrier
42h Lana Local Hot Carrier
43h Lana No Carrier Detected
44h unusual network condition
45h-4Fh hardware error
50h adapter malfunction
FFh NetBIOS busy (command pending)
```

Valores devueltos como código de retorno de una llamada a NETBIOS.

No se permiten reentradas al DOS. Por consiguiente, no se puede hacer ninguna llamada a sus funciones. Si se intenta lo único que se puede conseguir es el "cuelgue" del sistema.

Hay que salir activando las interrupciones. Para ello, lo mejor es definir la función como se muestra en la figura 5.

Después de que se ha recibido un datagrama, hay que hacer un "rearme" si se quiere seguir recibiendo información. El rearme consiste en volver a ejecutar una orden NbReceiveDatagram. Lo más aconsejable es realizar dicha operación desde fuera de la rutina ESR. En caso de hacerse desde dentro, hay que activar antes las interrupciones para poder establecer el enlace con NetBios, así como impedir mediante la utilización de alguna variable de estado la reentrada a la propia rutina ESR mientras no se salga de ella.

ENVÍO DE LOS MENSAJES INTRODUCIDOS POR EL TECLADO

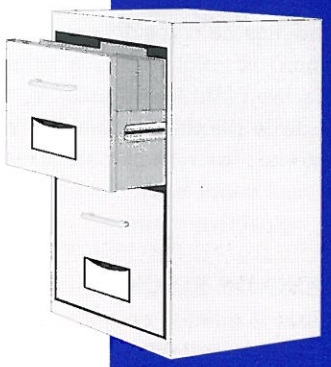
Como es lógico, se utilizan datagramas para enviar los mensajes introducidos a través del teclado. Cada vez que se pulsa la tecla <ENTER>, se lee la información introducida por el teclado y se envía a la estación remota. Para salir de la aplicación se pulsa <ENTER> dejando una línea en blanco.

CIERRE DE LA COMUNICACIÓN Y BORRADO DEL NOMBRE

Para cerrar la comunicación es suficiente con enviar una orden NbReset al NetBios. De esta manera se eliminan todas las órdenes pendientes. Lo único que queda es borrar de la tabla de nombres el nombre que se dio de alta al entrar en la aplicación. Si la variable de estado flagName es igual a cero significa que no se dio de alta ningún nombre en la tabla y por tanto no hay que borrar ninguno. ■

CA-CLIPPER 5.2 EL SISTEMA GET

Juan Manuel y Luis Martín



Con la aparición de la versión 5.00 del lenguaje Clipper, uno de los aspectos que más mejoró, fue la edición de campos y variables. Ésto se consiguió gracias a la inclusión en el lenguaje, de la programación orientada a objetos. Así, las tradicionales plantillas de edición de los mandatos @...GET, que pueden seguir utilizándose de la forma antigua, pasan a ser objetos de una nueva clase que aumenta sus posibilidades. Esta nueva clase, junto a una serie de recursos que también han sido incluidos, constituyen un completo sistema de edición, denominado Sistema GET. Este sistema se encuentra implementado en el archivo GETSYS.PRG situado en el directorio \CLIPPER5\SOURCE\SYS, y su código objeto es incluido de forma automática por el compilador.

Para utilizar el sistema GET en la edición de variables, ya sean de memoria o de campo, es necesario crear objetos a partir de la clase GET. Estos objetos representan las plantillas de edición que actuarán como interfaz entre el usuario y la variable. Por tanto, será necesario crear un objeto GET por cada variable que se desee editar. Para realizar el proceso de edición de varios de estos objetos en la misma pantalla, éstos deben ser añadidos a una estructura auxiliar después de ser creados. Esta estructura auxiliar es una matriz, que será pasada como argumento a la función encargada de realizar el proceso de edición propiamente dicho.

CREACIÓN DE OBJETOS DE LA CLASE GET

Los objetos de la clase GET pueden crease de forma explícita, o utilizando el mandato @...GET tradicional. Para crear explícitamente un objeto de la clase GET se utiliza el método constructor de la misma:

```
GETNEW( [nFila], [nColumna],
[bBloque], [idVariable], [cPicture],
[cColor] )
```

Esta función devuelve un objeto GET con las especificaciones que han sido indicadas en los argumentos de la llamada. Estos valores indican la posición de la plantilla de edición en la pantalla, el bloque de código que relaciona el objeto con la variable editada, el nombre de dicha variable y las opciones de formato y color de la plantilla.

Además de estos valores, es posible personalizar más aún los objetos GET. Para ello, deben asignarse valores directamente a sus atributos. La tabla 1 muestra una lista completa de los atributos de la clase GET.

Por ejemplo, para realizar la edición de un registro de una base de datos en una ventana como la mostrada en la figura 1, será necesario crear los objetos de la siguiente forma:

```
oTit := GetNew( 10, 10, FIELD-
BLOCK( "titulo" ),;
   "titulo", "@!", "B/W,R/BG" )
oAut := GetNew( 12, 10, FIELD-
BLOCK( "autor" ),;
   "autor", "@!", "B/W,R/BG" )
oEdi := GetNew( 14, 10, FIELD-
BLOCK( "edicion" ),;
   "edicion",, "B/W,R/BG" )
oTem := GetNew( 16, 10, FIELD-
BLOCK( "tema" ),;
   "tema", "@!", "B/W,R/BG" )
```

Este artículo estudia la utilización de objetos para mejorar los tradicionales mandatos de edición de campos y variables, @...GET y READ, permitiendo de este modo potenciar sus posibilidades y aumentar el control sobre el proceso.

Como puede observarse, es posible utilizar la función estándar FIELDBLOCK() para la especificación de los bloques de código que acceden a los campos. Para la edición de variables de memoria debe utilizarse en su lugar la función MEMVARBLOCK(). Estas funciones generan bloques de código de lectura y asignación del tipo siguiente:

```
{ | v | IF( v == NIL, idVar, idVar := v ) }
```

En las cadenas de color debe especificarse primero el color con que serán visualizadas las plantillas inactivas, y después el de la plantilla activa. En el ejemplo anterior, los mencionados colores serán azul sobre blanco y rojo sobre celeste, respectivamente.

EL PROCESO DE EDICIÓN

Para realizar la edición en una pantalla de un grupo de variables (por ejemplo, todos los campos de un determinado registro)deben seguirse varios pasos. Primero es necesario crear tantos objetos GET como variables vayan a ser editadas. Como ya se ha mencionado anteriormente, una vez creados, los objetos deben ser añadidos a una matriz de edición. Para el ejemplo anterior el código será el siguiente:

```
aGets := {}
AADD( aGets, oTit )
AADD( aGets, oAut )
AADD( aGets, oEdi )
AADD( aGets, oTem )
```

Finalmente, debe realizarse una llamada a la función del sistema GET encargada de realizar el proceso de edición, la función READMODAL(). Esta función recibe como argumento la matriz con los objetos que deben ser editados en la misma pantalla. Para el ejemplo anterior, la matriz pasada como argumento será la matriz aGets:

```
READMODAL( aGets )
```

Al comenzar el proceso de edición, en la pantalla se visualizará una plantilla por cada uno de los objetos contenidos en la matriz, con el formato y el color que se especificó para cada uno de ellos durante su creación.

Durante la edición, solo uno de los objetos estará activo, apareciendo el cursor dentro de su plantilla y pudiéndose alterar los datos de ésta. Se dice que el objeto que está activo es aquel que "tiene el foco". Las modificaciones que el usuario va realizando sobre la plantilla no se reflejan directamente en la variable, sino que se van almacenando en una zona de memoria intermedia, denominada buffer de edición. Esta zona está implementada en el atributo buffer de cada uno de los objetos como variable de tipo carácter.

Una vez finalizado el proceso, los cambios realizados en el buffer de edición pasarán a la variable. Es conveniente vaciar la matriz de edición antes de cargar el grupo de objetos que va a ser editado. Esta operación también suele realizarse después de cada llamada a la función READMODAL().

EL MANDATO @...GET

Como ya se ha mencionado anteriormente, Clipper permite seguir utilizando los mandatos @...GET y READ en la edición de campos y variables.

```
CLEAR
@ 10, 10 GET var1
@ 11, 10 GET var2
READ
```

Como resultado de la compilación se obtiene un archivo fuente con la extensión .PPO, que refleja los cambios realizados por el preprocesador en el programa fuente:

```
LOCAL var1, var2
Scroll() ; SetPos(0,0)
ReadKill(.T.) ; GetList := {}
SetPos( 11, 10 )
AAdd( GetList, _GET_( var1,
"var1",,, ):display() )
SetPos( 10, 11 )
AAdd( GetList, _GET_( var2,
"var2",,, ):display() )
ReadModal(GetList) ; GetList := {}
```

Como puede verse en las primeras líneas, el mandato CLEAR, además de borrar la pantalla, termina la edición de cualquier GET pendiente y vacía la matriz GetList. Cada uno de los mandatos GET sitúa el cursor en la posición especificada y añade a la matriz GetList el objeto adecuado. Los objetos no se crean con el constructor anteriormente descrito, sino mediante una función interna del sistema. Simultáneamente se invoca el método DISPLAY() del objeto, que se encarga de visualizar en la pantalla la correspondiente plantilla de edición. Finalmente, el mandato READ se traduce en una llamada a la función READMODAL(), vaciándose posteriormente la matriz GetList.

La creación de objetos GET puede realizarse explícitamente o mediante los mandatos tradicionales

Esto es posible gracias a la conversión al sistema GET realizada internamente por el preprocesador. Cada vez que éste se encuentre los mandatos @...GET y READ, se encargará de crear los objetos GET adecuados, añadirlos a la matriz de edición y llamar a la función que realiza el proceso de edición propiamente dicho.

En este caso, la matriz de edición es proporcionada por el propio sistema GET. Se trata de una matriz denominada GetList, que se crea automáticamente como variable pública al arrancar la aplicación y que, en principio, se encuentra vacía.

Para comprender este proceso, es conveniente analizar las conversiones realizadas por el preprocesador. Para ello, puede compilarse con la opción /P un sencillo programa que edite dos variables de memoria:

```
LOCAL var1, var2
```

LOS NIVELES DE SISTEMA GET

Mediante la utilización de los objetos de la clase GET, el programador puede escoger tres niveles diferentes de funcionamiento, dependiendo del control deseado sobre el proceso de edición. Además de un mayor control, la profundización en un nuevo nivel añade un mayor esfuerzo en la programación,

ya que se aumenta la complejidad del proceso. Por tanto, es recomendable utilizar el nivel más alto, siempre que éste proporcione el control suficiente sobre el proceso a realizar. Los tres niveles posibles son:

- Nivel read
- Nivel reader
- Nivel de funciones

El nivel más elemental es el nivel read. En este nivel, la edición de campos y variables se realiza mediante los mandatos @...GET y READ tradicionales. Por su parte, el nivel reader permite controlar el proceso de edición de cada uno de los objetos GET, es decir, permite rediseñar la rutina encargada de realizar dicho proceso. Por último, el nivel de funciones consiste en la realización completa del proceso de edición, haciendo uso directo de los distintos métodos de los objetos y de las funciones asociadas al sistema GET.

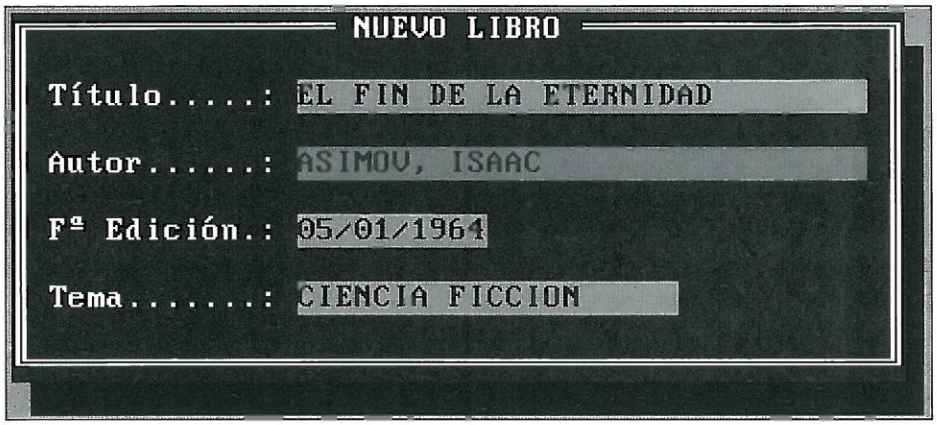


Figura 1: Edición de un registro nuevo de una base de datos.

EL NIVEL READ

Como se ya se ha mencionado anteriormente, los mandatos @...GET y READ tradicionales son convertidos internamente al sistema GET por el preprocesador. Esta conversión, igual que la del resto de mandatos del lenguaje, se realiza a través del fichero de cabecera STD.CH del directorio \CLIPPER5\INCLUDE. Este fichero es incorporado automáticamente a los programas por el compilador. En el cuadro 1 se muestran las operaciones realizadas por el preprocesador para la implementación del mandato @...GET.

Como puede observarse, la ejecución del mandato supone, primeramente, el posicionamiento del cursor en las coordenadas especificadas. A continuación, se crea el objeto GET con los valores especificados en las cláusulas PICTURE, VALID y WHEN. Para que la plantilla de edición sea visualizada en la pantalla, en la misma operación se ejecuta el método DISPLAY(). Finalmente, el objeto se añade al final de la matriz GetList.

En el nivel read, el proceso de edición se encuentra totalmente automatizado. Es el propio sistema el que se encarga de crear la matriz GetList, y de añadir a ésta los distintos objetos creados. El proceso de edición se activa con la llamada al mandato READ, que desencadenará las operaciones necesarias para producir la edición de los objetos GET incluidos en la matriz GetList.

A pesar de la simplicidad de éste nivel, es posible realizar algunas operaciones que mejoran el proceso de edición sin realizar demasiados cambios en el sistema.

READ ANIDADOS

Mediante la utilización del sistema GET, es posible realizar procesos de edición anidados. Se trata de interrumpir el proceso de edición en curso para realizar otro, de forma que al finalizar este último, el control vuelva al primero en el punto en que fue interrumpido.

Para realizar esta operación es necesario crear una segunda matriz de edición independiente. Mediante la asignación de una función de usuario a una tecla es posible interrumpir el proceso de edición de la primera matriz, de forma que dicha función cree una nueva matriz de edición local. Esta matriz será cargada con nuevos objetos GET, procediéndose entonces a su edición. Una vez finalizado el proceso, la ejecución de la función concluye y el control es devuelto al primer proceso de edición en el punto en que fue interrumpido.

Dentro del fichero cuadro 2, TXT SE muestra la forma de interrumpir la edición de dos campos de una base de datos para editar otros dos distintos. Para la realización de este ejemplo se ha utilizado el mandato SET KEY, asignando una función de usuario a la pulsación de la tecla [F2].

Como puede observarse en el listado, la función de usuario se encarga no sólo de preparar el segundo proceso de edición, sino que también debe preservar y restaurar aquellas partes de la pantalla que vayan a verse afectadas.

AMPLIACIÓN DEL MANDATO @...GET

Si se observa con detenimiento la implementación interna del mandato @...GET, mostrada en el cuadro 1, puede verse que se ha incluido una cláusula nueva al formato del mismo respecto a versiones anteriores. Se trata de la cláusula SEND, que permite enviar mensajes al objeto recién creado. De este modo, es posible ejecutar métodos del objeto GET, o bien asignar valores directamente a sus atributos.

```
@...GET...SEND método()
@...GET...SEND atributo := valor
```

La traducción que de esta cláusula realiza el preprocesador se pone de manifiesto en la siguiente operación:

```
ATail( GetList ):método()
ATail( GetList ):atributo := valor
```

La utilización de esta cláusula proporciona multitud de posibilidades, ya que permite alterar los valores por defecto de los atributos del objeto. Un claro ejemplo de ello es la inclusión de una cadena de color en el mandato @...GET, aunque éste no cuente con la cláusula COLOR para tal efecto. Bastará con asignar la cadena de color

Tabla 1

Atributo	Descripción
BadDate	Contiene un valor lógico que en caso de que la variable a editar sea una fecha indica si es válida o no.
Block	Contiene el bloque de código de lectura/escritura que relaciona el objeto con la variable o campo que se está editando.
Buffer	Es una variable de tipo carácter que contiene la información intermedia durante la edición.
Cargo	Es una variable de propósito general para almacenar cualquier información.
Changed	Contiene un valor lógico que indica si el contenido del buffer ha sido alterado después de "recibir el foco".
Col	Contiene un valor numérico que indica la columna donde aparecerá la plantilla de edición.
ColorEspec	Contiene una cadena de caracteres con el color a utilizar.
DecPos	Contiene un valor numérico que indica la posición del punto decimal dentro del buffer.
ExitState	Contiene un valor numérico que indica la forma en que se salió del proceso de edición. Sus valores se encuentran en la tabla 2.
HasFocus	Contiene un valor lógico que indica si el objeto actual "posee el foco".
Name	Contiene el nombre de la variable o campo que se está editando con ese objeto.
Original	Contiene el valor original de la variable cuando el objeto "recibió el foco".
Picture	Contiene una cadena de caracteres que indican el formato de la plantilla de edición.
Pos	Contiene un valor numérico que indica la posición del cursor dentro del buffer.
PostBlock	Contiene el bloque de código encargado de realizar la postvalidación del valor introducido.
PreBlock	Contiene el bloque de código encargado de realizar la prevalidación del valor introducido.
Reader	Contiene el bloque de código encargado de realizar el proceso de edición alternativo.
Rejected	Contiene un valor lógico que indica si el último carácter tecleado ha sido introducido en el buffer o no.
Row	Contiene un valor numérico que indica la fila donde aparecerá la plantilla de edición.
Type	Contiene una letra que indica el tipo de datos de la variable o campo que se está editando con ese objeto.
TypeOut	Contiene un valor lógico que indica si se ha sobrepasado la longitud del buffer.

Atributos de los objetos GET.

Tabla 2

Valor	Identificador	Descripción
0	GE_NOEXIT	No se ha intentado salir
1	GE_UP	Se pasó al GET anterior
2	GE_DOWN	Se pasó al GET siguiente
3	GE_TOP	Se pasó al primer GET
4	GE_BOTTOM	Se pasó al último GET
5	GE_ENTER	Salida normal pulsando la tecla [Intro]
6	GE_WRITE	Finaliza el READ con la variable actualizada
7	GE_ESCAPE	Finaliza el READ sin almacenar
8	GE_WHEN	No se cumplió la cláusula WHEN

Valores posibles del atributo ExitState.

Cuadro 1

```
/***
* @..GET
*/

#command @ <row>, <col> GET <var>      ;
                  [PICTURE <pic>] ;
                  [VALID <valid>] ;
                  [WHEN <when>]   ;
                  [SEND <msg>]    ;
                                  ;
 => SetPos( <row>, <col> )          ;
  ; AAdd(                          ;
    GetList,                       ;
     _GET_( <var>, <(var)>, <pic>,  ;
      <{valid}>, <{when}> ):display();
       )                           ;
  [; ATail(GetList):<msg>]
```

Definición del mandato @...GET.

deseada al atributo ColorEspec, utilizando para ello la cláusula SEND.

```
@...GET...SEND ColorEspec :=
"W/B,G/R"
```

La utilización del atributo Cargo junto con la cláusula SEND permite ampliar la información contenida en el objeto durante su creación. Esta información puede ser utilizada posteriormente durante el proceso de edición del objeto con el fin de mejorarlo. En el fichero cuadro 3. TXT se muestra un programa ejemplo en el que se hace uso del atributo Cargo para almacenar el texto de una ayuda on-line relativa a cada objeto.

Como puede observarse, el proceso de edición es interrumpido al pulsar la tecla [F1], ejecutándose una función de usuario. Esta función se encarga de visualizar un cuadro con el contenido del atributo Cargo del objeto GET activo, haciendo uso para ello de la función MEMOEDIT(). Al igual que en el ejemplo anterior, es la función de usuario la encargada de salvaguardar las zonas de la pantalla que van a verse afectadas.

Para la realización de este ejemplo, ha sido necesario utilizar una de las funciones del sistema GET. Se trata de la función GETACTIVE(), que devuelve el objeto GET que está activo, es decir, el que tiene el foco.

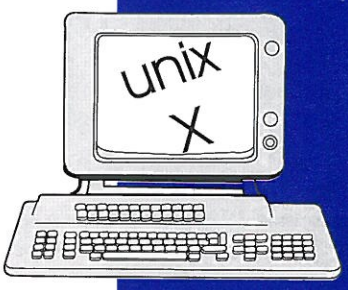
Otra forma de ampliar el mandato @...GET consiste en la utilización de las cláusulas WHEN y VALID. En las últimas versiones del lenguaje, no sólo es posible indicarlas como expresiones lógicas, sino que también pueden ser especificadas en forma de bloques de código. Estos bloques de código se almacenan en los atributos PreBlock y PostBlock, respectivamente. De esta forma, es posible ejecutar funciones de usuario cuando el objeto "recibe o cede el foco" durante el proceso de edición. Es importante recordar que estos bloques de código deben devolver el valor lógico apropiado para permitir la entrada y salida del proceso de edición.

CONCLUSIÓN

El nuevo sistema para la edición de campos y variables de memoria mediante la utilización de objetos, permite al programador diseñar procesos de edición personalizados e incrementar el control sobre los mismos. Sin embargo, como ha podido observarse a lo largo del artículo, sigue siendo posible utilizar los mandatos @...GET y READ de la forma clásica. Además, la nueva implementación de éstos permite realizar procesos de edición más completos sin necesidad de grandes cambios en la forma de trabajo. ■

FILTROS UNIX

Fernando J. Echevarrieta



En los capítulos precedentes se realizó la exposición de los elementos básicos del sistema. Sin embargo, debido a la gran amplitud del mismo, aún no se podía realizar gran cosa con ellos, salvo moverse con timidez de un lado a otro. El pasado mes se describió el uso del editor vi con vistas a disponer de una herramienta básica para comenzar a modificar el entorno.

A partir de este capítulo, paralelamente al contenido principal de cada artículo, se desarrollará una serie de conceptos prácticos, trucos o formas de configuración que, aunque en el futuro sean explicados con más precisión, es interesante y, a veces, necesario, comenzar a manejar desde ahora. Así, en el presente artículo se tratará la utilización, formato y montaje de disquetes desde Linux, y cómo saltar entre consolas virtuales. El grueso del capítulo presentará el potente sistema de filtros UNIX, herramienta básica para la construcción de scripts de la shell, aunque por esta vez, sólo se tratarán desde línea de comando.

Existe una clase especial de comandos UNIX llamados filtros

COMANDOS Y FILTROS

Los comandos del sistema suelen implementarse en ejecutables binarios que se encuentran en directorios destinados al efecto (/bin, /sbin, /usr/bin, /usr/etc, /usr/etc, /usr/ucb, /usr/local/bin, etc), aunque algunos de ellos pueden ser internos a la shell con que se trabaja como ocurre en DOS. Entre estos comandos, que no dejan de ser programas, se encuentra un tipo especial llamado filtro, que se caracteriza por las siguientes propiedades:

1. Toma los datos siempre de la entrada estándar (STDIN), lo que le dota de una gran versatilidad ya que, al no leer de un dispositivo concreto sino de un descriptor redirigible, permite convertir prácticamente cualquier cosa en su fuente de información.
2. Los resultados siempre son dirigidos a la salida estándar (STDOUT), con las mismas ventajas que se citaban en el párrafo anterior, esta vez, con la salida.
3. Realizan una función de transformación de la entrada y no tiene lugar ningún otro proceso, por lo que como todo filtro se rige por la ecuación: salida = f(entrada) exclusivamente.
4. Por el principio anterior, salvo excepciones, se trata de comandos no interactivos. En el caso de las excepciones, se trata de una interactividad que no afecta a la función de proceso del filtro (por ejemplo, en un paginador).
5. Los errores se presentan a través de la salida de error estándar (STDERR), por lo que se puede mantener un flujo de error aparte con las ventajas del punto 2.

Son estas características especiales, las que dotan al interfaz de comandos UNIX de su plena potencia con la incorporación del elemento PIPE: mediante la interconexión en cadena de filtros, a través de pipes se puede crear un número virtualmente ilimitado de nuevos comandos, con funciones de proceso cada vez más complejas que

Una vez se dispone de la base necesaria para la navegación por el sistema y del manejo de un editor, es el momento de presentar el conjunto de filtros UNIX con vistas a su inclusión en scripts así como comenzar a utilizar algunos "trucos" del Linux.

derivarán de la composición de las funciones de proceso de los componentes. En la figura 3 se puede observar esta cualidad. Si UNIX proporcionara un filtro que dada una vaca obtuviera leche y otro que dada leche obtuviera queso, ambos se podrían interconectar en línea de comando o formando parte de un programa para realizar una transformación de leche en queso. UNIX, en realidad, proporciona un potente juego de filtros para actuar sobre ficheros, lo que en el mundo UNIX quiere decir sobre "todo". Hay que destacar una vez más que si bien en DOS se necesitaría obtener toda la leche para hacer queso (ver figura 2), en UNIX se comienza a obtener el queso desde que se obtiene la primera gota de leche ya que ambos filtros funcionarán concurrentemente.

El rango de complejidad de los filtros abarca desde los más simples hasta los más complicados programas

¿CÓMO FUNCIONA POR DENTRO?

En general un filtro se puede invocar de dos formas: con y sin argumentos. En el caso de llevarlos, por ejemplo, sort /etc/passwd, es el programa-filtro el que abre el fichero, por lo que en realidad no se comporta como tal ya que no lee de la entrada estándar. Sin embargo, el programa puede invocarse como sort < /etc/passwd o como cat /etc/passwd | sort. En estos dos casos, será la shell la encargada de abrir el fichero y sí existirá comportamiento de filtro. Para una correcta comprensión del sistema es importante tener en cuenta siempre que se trabaja en un entorno multiprogramado.

UNA EXCEPCIÓN AL EJEMPLO GENERAL

Pese a lo que en principio pueda parecer, y a pesar de tratarse de programas interactivos, los paginadores more y less pertenecen al conjunto de los filtros. En este caso, el proceso consiste únicamente en el formateo de la entrada para una correcta representación en un terminal. Su sintaxis responde a:

more [opción] [fichero]
less [opción] [fichero]

Su característica de filtro, les permite ser redirigidos (aunque no es característica exclusiva), con lo que se puede mostrar su salida a través de un terminal distinto de aquel desde el que se han lanzado. Así, mediante less /etc/passwd > /dev/tty1a, la salida aparecerá en el terminal indicado, que puede ser el que está utilizando otra persona siempre que el ejecutante disponga de permiso de escritura (ya sea de usuario, grupo, u otro como se explicó en un número anterior). Ésta es una anecdótica forma de tomar el control de otro terminal, ya que el filtro aún tendrá asociada su entrada (y control) al terminal desde el que se ejecutó).

```
#! /bin/sh
for i in *
do
mv $i `echo $i | tr A-Z a-z`
done
```

Listado 1.

La interactividad, por otra parte, no afecta al proceso de la entrada. Así entre los comandos que aceptan, se puede citar: q, Q, provocan la terminación; h, muestra ayuda; v, invoca a vi sobre el documento mostrado; !, invoca una shell; y los de movimiento, como ENTER, SPACE, etc.

La shell dispone de una variable llamada PAGER, a la que se puede asignar el nombre de un paginador y que será el utilizado por defecto para mostrar textos en programas como man, mail, lectores de news, etc.

EJEMPLOS DE FILTROS SIMPLES

Otros filtros simples utilizados habitualmente son:

head [-n] [fichero]: Muestra las n primeras líneas de "fichero".

tail [pto comienzo] [fichero]: Muestra las últimas líneas de "fichero", desde el punto de comienzo que puede ser +n (línea n-ésima desde el comienzo del fichero) o -n (línea n-ésima desde el final). En el ejemplo 1 se puede ver una sencilla combinación que muestra el último fichero modificado. El comando ls presenta una salida en orden cronológico (opciones -rt) y el comando tail extrae el último elemento.

strings [fichero]: Presenta las cadenas de texto contenidas en un fichero, lo que puede ser muy útil a la hora de buscar en ficheros binarios.

cat [fichero]: Simplemente presenta el fichero en pantalla.

uniq: Elimina las líneas repetidas.

compress [fichero]: Es el compresor estándar de UNIX. En el caso de aplicarse sobre un fichero, produce como salida otro del mismo nombre pero terminado en .Z (hay que recordar que en UNIX no existe el concepto de extensión de un fichero). Es importante destacar que el fichero original desaparece.

uncompress [fichero]: Filtro inverso de compress.

Los filtros han sido concebidos para concatenarse entre sí componiendo nuevas funciones

Además de estos filtros que se podrán denominar "originales", también se han ido incorporando otros que no se encontraban presentes en el UNIX original, del mismo modo que ocurre con los compresores y utilidades DOS. Así, por ejemplo, casi siempre (es el caso de Linux), se dispone de un compresor zip (el gzip).

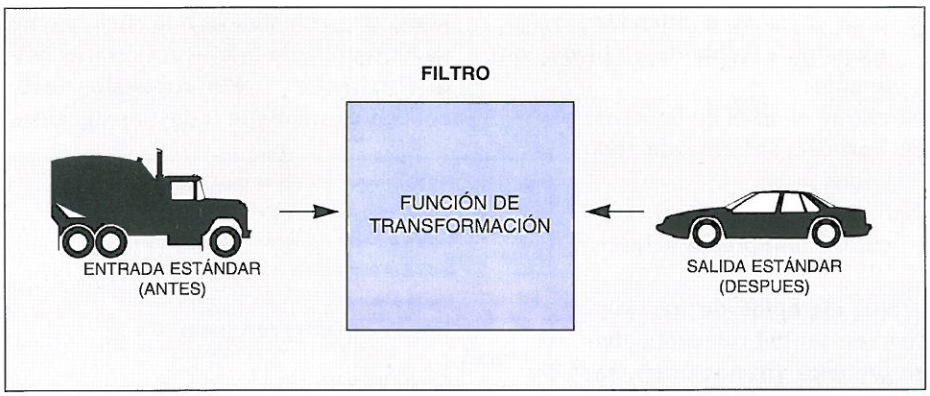


Figura 1. Funcionamiento de un filtro unix.

TRADUCCIÓN DE CARACTERES

Un filtro muy interesante es el comando tr, cuya función de proceso constituye una transliteración de la entrada. Se invoca mediante: tr rango.original rango.final. Así, el ejemplo 2, pasaría todos los caracteres de la STDIN de mayúsculas a minúsculas, y el ejemplo 3 crearía un fichero file2 con el contenido de file1 pasado a minúsculas. La verdadera potencia de los filtros vendrá dada cuando se incluyan en scripts de shell.

En el listado 1 se puede apreciar cómo un filtro tan simple como éste, puede servir para realizar un programa que cambie los nombres de los ficheros de mayúsculas a minúsculas, que puede ser interesante, si utilizamos discos remotos montados por NFS desde DOS.

Otra pareja de útiles comandos son expand y unexpand.

El primero de ellos sustituye ocho caracteres en blanco (por defecto) por un tabulador mientras que el segundo realiza la operación inversa. Así, el ejemplo 4 reemplazará cada carácter tabulador por 6 caracteres en blanco.

ORDENACIÓN Y BÚSQUEDA

Para poder realizar scripts en shell que se precien de ser auténticos programas, no basta con los filtros expuestos hasta el momento. Es necesaria la combinación de un potente mecanismo de ordenación y búsqueda, para disponer de una base con la que empezar a construir. Para la ordenación, UNIX provee el filtro:

sort [opciones] [posición...] [fichero]

donde se utilizará la entrada estándar, y si el nombre de fichero es -o se omite. La posición indica el segmento de texto que se comparará para ordenar y tiene la forma: +pos1[-pos2] siendo pos1 y pos2 las posiciones de comienzo y fin de la cadena respectivamente. Cada una de estas posiciones puede expresarse como f[.c] donde f=field (campo) y c=char (carácter). Así +2 -3, representa el tercer campo; +5, toda la línea desde el sexto campo, y +2.3 -3.4, desde el cuarto carácter del tercer campo hasta el quinto del cuarto campo.

Para filtrar búsquedas, UNIX proporciona la familia de comandos grep. Las diferencias entre unos y otros son su velocidad, requerimientos de memoria y patrones de búsqueda. En general, para búsquedas que no requieran expresiones regulares, el comando más rápido será fgrep. Se les invoca según la sintaxis siguiente:

fgrep [opciones] cadena [fichero]
grep [opciones] expr regular [fichero]
egrep [opciones] expr regular [ficheros]

donde las opciones y expresiones regulares más usuales vienen recogidas en las tablas 1 y 2. El ejemplo 5 extrae las cadenas de texto que comiencen por p y terminen por s del fichero cap6.wp. El ejemplo 6 busca campos vacíos en el /etc/passwd y el 7, exige que el campo vacío sea el correspondiente al propio passwd (una forma como cualquier otra de empezar a atacar un sistema). Los ejemplos 8 y 9 buscan las apariciones de las cadenas "Sólo" o "Programadores" en STDIN. En el primero de los casos con cualquier combinación de mayúsculas y minúsculas, y en el segundo permitiendo la combinación sólo en la letra inicial de la palabra. Por último, el ejemplo 10 busca la cadena ASCII "teléfono" en cualquier combinación de mayúsculas y minúsculas en un fichero ejecutable de DOS, lo que puede ser una idea para encontrar al autor.

.	
*	
[lista]	
[^lista]	
^	
$	Cualquier letra
La cadena precedente 0 o + veces	
Alguno de los caracteres en lista	
Ninguno de los caracteres en lista	
Principio de línea	
Final de línea	

Tabla 1: expresiones regulares de la familia GREP.

PERO...¿QUÉ PASA CON LAS OPCIONES?

En el primer punto se afirmaba que la función de transformación de un filtro dependía únicamente de la entrada. Sin embargo, en los filtros más complejos que se han expuesto se puede

El conjunto de filtros UNIX proporciona potentes mecanismos de ordenación y búsqueda

A través de un comando tan simple como less se puede tomar el control de otro terminal

apreciar que pueden ir acompañados de opciones que harán que el filtro se comporte de una forma o de otra. Esto puede inducir a pensar que se ha producido una contradicción: no es así. Del mismo modo que se justificaba la existencia de filtros con cierta interactividad, ya que ésta no variaba la función de transformación del filtro, en el caso de las opciones ocurre algo similar. Es cierto que dadas distintas opciones se obtienen distintas funciones de transformación pero una vez se invoca un filtro con unas opciones, el comportamiento queda fijado y la salida únicamente dependerá de la entrada. Esto equivale a disponer de un nombre distinto para cada transformación. En lugar de ello, si las transformaciones son muy parecidas, se les da un nombre común y se las diferencia por las opciones. Volviendo al ejemplo de la vaca en la figura 3, se puede apreciar que el segundo filtro

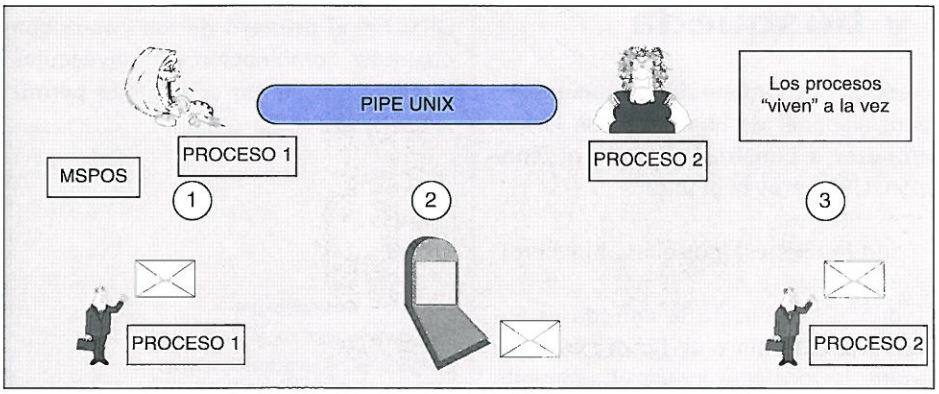


Figura 2. Comunicación entre pipes.

esta conectado a una serie de opciones. El filtro podría llamarse "procesa_leche" y en función de la opción que se le indicara obtener cuajada, yogures o queso.

EL COMANDO TAR

El comando tar (Tape ARchiver), nació con la idea de realizar backups en cinta, como su nombre indica. Su funcionamiento consiste en reducir un árbol de ficheros a un solo fichero que pueda ser almacenado con facilidad. Por tanto, no se trata de un compresor como el gzip. Se le invoca mediante:

tar [opciones [parámetros de opción] [árbol]

Las opciones principales de tar son:

-c: Crea un nuevo archivo tar.
-t: muestra la Tabla de archivos contenidos.
-x: eXtrae el árbol de ficheros.
-v: Verbose. Informa de todo lo que hace.
-f: Fichero. Tras ella se indicará el nombre de fichero.

Así, ejemplos de uso pueden ser: tar cvf micuenta /home/echeva (almacenar), tar tvf micuenta (ver contenido), tar xvf micuenta (recuperar).

UN PAR DE EJEMPLOS "POTENTES"

El ejemplo 11 procesa todos los ficheros cuyo nombre encaja con el patrón unix[cadena].txt, generando un fichero llamado lista que contiene todas las palabras y números contenidos en los mismos (sin duplicar) indicando su frecuencia y muestra por pantalla el número de palabras encontradas. El comando tr se encarga de colocar una palabra por línea al sustituir todos los caracteres no alfanuméricos (opción -c, los caracteres que no coinciden; opción -s, sin repeticiones) por avances de línea (carácter 012). Tras ello, se ordena alfabéticamente (sort) y se eliminan las líneas duplicadas (uniq) indicando en cada una su frecuencia de aparición (opción -c). El comando tee es un duplicador de "flujos", que repite su entrada por dos caminos: la salida estándar y un fichero que se le indique, en este caso "lista". Tras ello, simplemente se cuentan las líneas y se muestra el resultado en pantalla. El ejemplo también sirve para ilustrar el hecho de que, si hay que escribir en más de una línea, se puede indicar a la shell que no se ha terminado la instrucción mediante el carácter '\'. Por supuesto, nadie conoce de memoria todas las opciones

-i	No tiene en cuenta si es mayúscula o minúscula
	Líneas que no coinciden con patrón
-v	Indica número de líneas que coinciden
-c	Indica el número de la línea que coincide
-n	Modo silencioso
-s	Toma las expresiones del fichero
-f fichero	

Tabla 2: principales opciones de la familia GREP.

de los filtros. El truco consiste en recordar las funciones principales, por ejemplo, que tr traduce caracteres. Cuando se desea realizar algo parecido, se consulta la página de manual del correspondiente filtro.

El ejemplo 12 supone una utilidad más sencilla, pero más espectacular y más útil. Consiste en un comando que envía por mail al autor del artículo el contenido del directorio en que se ejecuta, y toda la estructura de sus subdirectorios en forma comprimida. El comando tar se encarga de la compactación en un sólo fichero de toda la estructura pendiente de . (directorio actual) y se comporta como un filtro (empleando STDIN) debido al uso de la opción -. Tras ello, el flujo de información se comprime y se codifica en formato ASCII para poder ser enviado a través de mail por Internet con el protocolo SMTP (ver capítulo 3 del curso).

Una vez se decodifique en el otro extremo, recibirá el nombre de dir.tar.Z que implícitamente indicará la forma de recuperar la estructura de directorios (consultar el capítulo 4 del curso en lo referente al mail). Los pasos a seguir en el otro extremo serían: 1. Salvar el mensaje; 2. Eliminar todas las líneas previas a la que comienza con BEGIN; 3. Ejecutar uudecode sobre el fichero salvado; 4. Descomprimir (uncompress dir.tar); 5. Extraer la estructura de directorios (tar xvf dir.tar).

HERRAMIENTAS Y TRUCOS LINUX

A partir de este capítulo, comienza esta breve subsección destinada a la aplicación de algunos conceptos, trucos o técnicas de administración que

CORREO LECTORES

Para cualquier duda referente a los temas/artículos tratatos en la revista, escriba una carta a:

Sólo Programadores
Referencia: *Correo Lectores*
TOWER COMMUNICATIONS
C/ Marqués de Portugalete, 10 Bajo
Madrid 28027

COMO SUSCRIBIRSE A

Suscríbase enviando este cupón por correo o fax (91) 320.60.72, o llamando al teléfono (91) 741.26.62 Horario 9 a 14 y 15:30 a 18:30 h.

Deseo suscribirme a la revista SÓLO PROGRAMADORES acogiéndome a la siguiente modalidad:

☐ Suscripción: 1 año + regalo (filtro monitor) **por sólo** 10.995 ptas. ☐ Estudiantes carreras técnicas 40% Dto.: 8.250 ptas.

Nombre y apellidos.....Domicilio.....

Población.....C.P .....Provincia.....Telf.....Profesión.....

FORMA DE PAGO:

☐ Con cargo a mi tarjeta VISA nº .....

Fecha de caducidad de la tarjeta.....Nombre del titular, si es distinto.....

☐ Domiciliación bancaria.
Señor Director del banco .....
Población .....

CODIGO CUENTA CLIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

Ruego a vd. que se sirva cargar en mi ☐ cuenta corriente ☐ libreta de ahorro el recibo que le será presentado por TOWER COMMUNICATIONS, S.R.L. como pago de mi suscripción a la revista SÓLO PROGRAMADORES.

☐ Contra-reembolso del importe más gastos de envío.
☐ Cheque a nombre de TOWER COMMUNICATIONS S.R.L., que adjunto.
☐ Giro Postal (adjunto fotocopia del resguardo).

Firma:

Rellena este cupón y envíalo a:
TOWER COMMUNICATIONS SRL
C/ Marques de Portugalete 10, Bajo
28027 Madrid.

permitan de una manera rápida, resolver ciertas necesidades, o disponer de determinadas capacidades del sistema que, por su importancia, deberían ser utilizadas antes de poder ser explicadas en detalle en el cuerpo principal del curso.

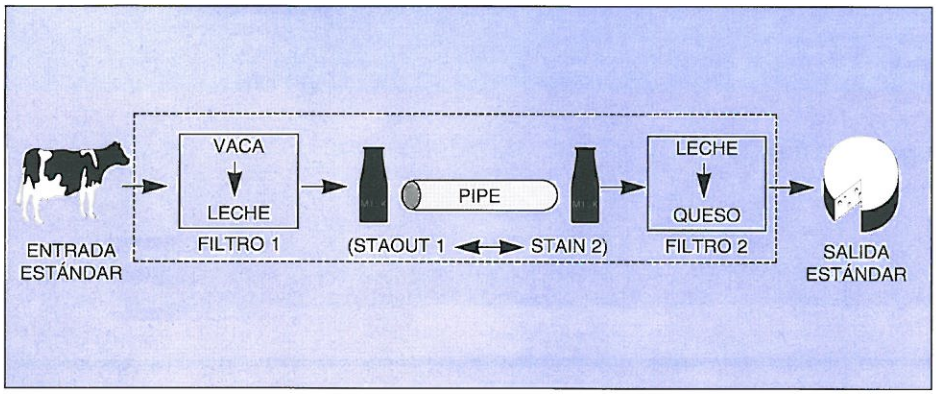


Figura 3. Asociatividad de filtros. A través de las opiniones se puede determinar la función de transformación del filtro 2.

FORMATEO DE UN DISQUETE LINUX

Como se comentó en el capítulo introductorio y como más adelante se verá en la parte dedicada al sistema de ficheros, para UNIX todo son ficheros, incluyendo los dispositivos, y el sistema de ficheros esta formado por volúmenes que se "montan" en un punto de anclaje. En esta breve sección que comienza hoy, se comenta cómo acceder a las disqueteras de un sistema Linux. Todo sistema UNIX suele contar con un directorio /mnt destinado a servir como punto de anclaje para montajes provisionales que, en este caso, se utilizará para montar las disqueteras. Así como en DOS se asocia la unidad lógica A: a la primera unidad de disquete y la unidad lógica B: a la segunda, en Linux se les asocia los ficheros especiales de dispositivo /dev/fd0 y /dev/fd1 respectivamente. Así, para generar un sistema de ficheros minix propio de Linux en un disquete en la unidad 0 (A:), suponiendo que sea de alta densidad y 3.5", por ejemplo, habrá que teclear:

```
mkfs /dev/fd0 1440
```

lo que equivaldría a teclear en DOS

```
FORMAT A:
```

```
1. ls -lrt | tail -1
2. tr A-Z a-z
3. tr A-Z a-z < file1 > file2
4. expand -6 file
5. strings unix6.wp | grep '^p*s$'
6. grep :: /etc/passwd
7. grep '^[^:]*::' /etc/passwd
8. egrep -i 'Sólo | Programadores'
9. egrep '[Ss]ólo | [Pp]rogramadores'
10. strings SETUP.EXE | grep -i teléfono
11. cat unix*.txt | tr -sc A-Za-z0-9 '\012' | \ sort | uniq -c | tee lista | wc -l
12. tar -cvf - . | compress | \
    uuencode dir.tar.Z | \
    mail echeva@dit.upm.es
```

Tabla de ejemplos con filtros.

UTILIZACIÓN DE UN DISQUETE LINUX

Para montar un disquete con sistema de ficheros minix (propio de Linux) en el directorio /mnt se deberá teclear:

```
mount /dev/fd0 /mnt
```

A partir de ese momento, todos los accesos al directorio /mnt se realizarán sobre el disquete. Se recuerda al lector que, como se indicaba en el capítulo 2, el sistema puede realizar una escritura en diferido, por lo que para que los cambios se hagan efectivos en el disquete habrá que teclear la orden sync. Del mismo modo, no deberá NUNCA extraerse el disquete de la unidad sin antes haber desmontado el dispositivo mediante la orden

```
umount /mnt
```

esta orden se encargará de reflejar los cambios como si se hubiera tecleado sync anteriormente.

UTILIZACIÓN DE UN DISQUETE DOS

Para utilizar un disquete formateado desde DOS, bastará con indicarlo mediante la opción -t:

```
mount -t msdos /dev/fd0 /mnt
```

Esta opción es propia de Linux ya que sobre otro S.O. como FreeBSD, el sistema de ficheros puede recibir otro nombre como pcfs. Si está siguiendo el curso con un S.O. UNIX distinto de Linux, consulte el manual referente a mount.

CONSOLAS VIRTUALES

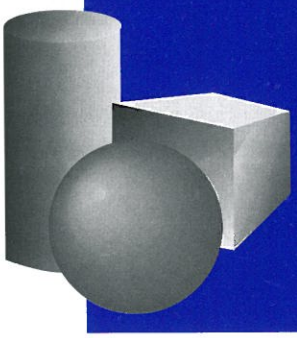
Linux permite la conmutación entre consolas virtuales en modo texto pulsando simultáneamente las teclas ALT y una tecla de función que indica el número de la consola, desde X se realizará pulsando simultáneamente CTRL+ALT+Fn. La consola virtual asignada por defecto en Linux al sistema X es la número 6.

EL PRÓXIMO CAPÍTULO

En el próximo capítulo se presentará el sistema de ventanas estándar de UNIX: X WINDOW. Se expondrán las distintas versiones que se disponen para PC y en concreto las existentes para Linux. Se hará un repaso al funcionamiento de las mismas y, tras contar su forma de instalación así como su configuración y personalización, se hará repaso de algunos trucos para sacar el máximo partido del sistema en equipos lentos. ■

GUÍA DE ESTILO C++

Ignacio Cea



Esta última entrega del curso de C++ pretende ofrecerles tanto una breve guía sobre los aspectos más relevantes de este potente lenguaje como unos consejos básicos sobre estilo de programación habitual en C++.

Como ya bien sabe, el C++ es un lenguaje basado en el C al que se le han añadido posibilidades para el trabajo con clases. Esta guía, por tanto, no va a hacer hincapié en aquellas características que son comunes a ambos, como formas de comentar, bucles o sentencias de control, sino sólo en las que los diferencian; esto es, las referentes al manejo de objetos.

Es conveniente emplear ficheros con extensión .hh y .cpp o .cc para conseguir mayor claridad

La guía rápida está contenida, básicamente, en las láminas coleccionables que acompañan a su revista. Las normas de estilo incluidas en la revista, están basadas, tanto en la experiencia del autor, como en las existen dentro de prestigiosas firmas como IBM o HP.

OBJETIVOS DEL C++

1.- El objetivo básico del C++ es facilitar la legibilidad y mantenimiento de sus aplicaciones, además de intentar aumentar la reusabilidad del código creado.

2.- Los programas C++ suelen ser algo más lentos que sus homólogos C. Sin embargo, pueden adaptarse con mucha mayor rapidez y claridad a cualquier cambio que se pretenda introducir en ellos.

3.-El tiempo de análisis y diseño de una aplicación C++ y, en general, de cualquiera orientada a objetos, debe ser mayor que el de una C. Dicho tiempo se verá ampliamente recuperado a la hora de reutilizar, en otras aplicaciones, el código escrito.

4.- En una aplicación C++ ha de primar la capacidad de abstracción del analista. Sólo así conseguirá sacar el máximo partido a la, tan pregonada, reusabilidad.

5.- El número de ficheros que integran una aplicación C++ es, normalmente, mayor que el de una equivalente C.

FICHEROS C++

1.- Para definir una clase se emplean dos tipos de ficheros: Los de cabecera y los de cuerpo.

2.- En los primeros se declara la estructura de clase y sólo los cuerpos de aquellos métodos declarados como inline.

3.- En los segundos se definen los cuerpos del resto de los métodos y se inicializan los atributos estáticos.

4.- Los ficheros de cabecera tienen que hacer uso del preprocesador

Para iniciarse en cualquier lenguaje es siempre conveniente disponer tanto de una guía rápida, que muestre los aspectos más relevantes del lenguaje, como de unas normas básicas sobre el mejor estilo de programación. Estas son las de Sólo Programadores para el C++.

```
// Fichero cabecera (.hh)
#if !defined (__CABECERA_HH__)
#define __CABECERA_HH__

class Clase {
        ...
        tipo METODOA (parámetros);
        inline tipo METODOB (parámetros);
        ...
};
...
inline tipo METODOB (parámetros) {
        ...
}
...
#endif

// Fichero cuerpo (.cc ó .cpp)
...
tipo CLASE::variable = ...;
const tipo CLASE::constante = ...;
...

tipo CLASE::METODOA (parámetros)
{
        ...
}
...
```

para garantizar que las clases en ellos contenidas no se declaran dos veces. Es habitual que en la compilación de un fichero .cc se invoque el mismo fichero include, aunque sea de manera indirecta. Si no evita que las clases de esos ficheros include se redefinan tendrá problemas de compilación.

5.- Los ficheros de cabecera suelen tener la extensión .hh y los de cuerpo .cpp o .cc. El motivo de esto es darles entidad propia y distinguirlos de ficheros C. No es obligatorio y, por supuesto, que puede usar las que mejor le convengan.

DEFINICIÓN DE CLASES

1.- Para declarar una clase se emplea la palabra clave class seguido del nombre de la clase y, entre llaves, la definición de los miembros de la misma. Tras la última llave ha de haber un punto y coma.

2.- Los miembros de la clase son, por defecto, privados. Si se desea que los miembros tengan otros identificadores de acceso debe hacer uso de las palabras clave public, private o protected.

```
class clase {
        public :
        ...
        protected :
        ...
        private :
        ...
};
```

3.- Aunque el C++ lo permite, los atributos de una clase deben ser siempre privados o protegidos con el fin de asegurar la encapsulación de la información (principio de la orientación a objetos). El acceso a los mismos debe hacerse, por tanto, a través de métodos. Hay, incluso, guías de estilo, como la de IBM, que desaconsejan también el empleo de atributos protegidos.

```
#if !defined (__CLASEA_HH__)
#define __CLASEA_HH__

#include "ClaseC.hh"
class ClaseB;

class ClaseA {
        ...
        private :
                ClaseB *objetoB;
                ClaseC objetoC;
                ...
        ...
};

#endif
```

Conviene que los atributos de una clase sean privados o protegidos

4.- El orden aconsejado para el uso de los identificadores de acceso es public, protected y private; es decir, de menor a mayor restricción en el ámbito de vista.

5.- Los métodos inline deben ser declarados dentro del fichero cabecera, pero siempre detrás de la definición de la clase a la que pertenecen.

6.- No use la palabra clave inline con un método friend, ya que no todos los compiladores lo admiten. Aplíquese lo mismo para los métodos static. BorlandC++ es un claro ejemplo de ello.

Los métodos inline deben ser declarados dentro del fichero de cabecera

7.- Todas los objetos que sean definidos como atributos de una clase, necesitan que su clase haya sido previamente declarada. Si ese empleo se hace sólo como puntero o referencia bastará simplemente con una declaración forward, mientras que si el uso es de cualquier otro tipo ha de invocar al fichero include que la contiene.

8.- El número de líneas de código ejecutable dentro de un método cualquiera de una clase no debe exceder de 50 o 60 (límite impuesto por el paquete para el control de la calidad del software QAC++). De otra manera, la legibilidad de su código se verá seriamente dañada. Siempre que le suceda eso busque, o bien la creación de un nuevo método dentro de la misma clase, o bien la construcción de una nueva clase que asuma parte de la responsabilidad del otro.

De entre todos los consejos destacan, sin duda, el cuarto y el quinto ya que, hasta el momento, ninguno de los ejemplos del curso ha seguido alguno de ellos pues se estimó que era más conveniente para el aprendizaje hacerlo de la otra manera.

Cualquier clase no hija de otra, só-

lo puede emplear los métodos public de aquella. Sólo por este motivo, es conveniente, declararlos al principio de la clase, pues es lo primero que leerá de ella un usuario.

Si se sigue la anterior norma y se usa un método inline dentro de la definición de la estructura para devolver, por ejemplo, el valor de un atributo que aún no ha sido definido (ya que será private y se declarará, por tanto, debajo del método inline que es public) se generaría un error de compilación, de ahí que sea conveniente declararlos detrás de la definición de la clase.

Emplee las directivas #ifndef, #define y #endif en los ficheros cabecera

CONSTRUCTORES Y DESTRUCTORES

1.- El C++ provee para cada clase tanto un constructor por defecto, como un copy constructor. El primero inicializa todo a 0, mientras que el segundo copia bit a bit el objeto parámetro.

```
ClaseA::ClaseA (parámetros)
                        : Objeto1 (),
Objeto2 (parámetros),
                                ...
{
        ...
}
```

2.- Es aconsejable, por razones de legibilidad, mantenimiento y funcionalidad, definir para cada clase ambos constructores, aunque se limiten bien a colocar todos los miembros a 0, bien a copiar uno a uno todos los miembros.

```
class ClaseA : public Padre {
        ...
        ~ClaseA () { /* Vacio */ }
};
```

3.- Inicialice, siempre que pueda, todos los atributos de una clase en la lista de inicialización del constructor. Es más rápido, pues evita hacer dos veces la inicialización de los mismos; si bien es cierto, que algunos compiladores ya optimizan este proceso.

4.- No ponga destructor a una clase que no tiene nada que destruir. No olvide, sin embargo, que el destructor es siempre necesario tanto para borrar la clase padre (si es que tiene algo que destruir) como para eliminar los objetos atributo definidos en el interior. En este caso es necesario colocar uno.

5.- Los constructores deben estar sobrecargados para conseguir la máxima versatilidad posible en la inicialización de un objeto. Los constructores juegan un papel muy importante en la legibilidad de un programa. Es decir, por ejemplo, el constructor de una supuesta clase rectángulo debería estar preparado, por ejemplo, para admitir como parámetros de inicio bien dos objetos punto, bien 4 coordenadas, etc...

6.- Algunos objetos miembros de una clase basta con que sean inicializados a través de sus constructores por defecto. A pesar de ello, invóquelos explícitamente, facilitan la legibilidad y no retardan en nada su aplicación.

7.- No es posible, ni tiene sentido, declarar un constructor virtual; al contrario de lo que sucede con los destructores.

8.- No declare constructores y destructores no públicos. No tiene sentido hacerlo.

```
...
Clase operator - (void) const;
...
```

SOBRECARGA DE OPERADORES

1.- Los operadores se sobrecargan sólo para conseguir un código más legible y fácilmente mantenible. No hay ninguna otra razón, ya que su funcionalidad bien podría conseguirse a través de otros tipos de métodos internos.

```
...
Complejo operator + (float) const;
friend Complejo operator + (float,const
Complejo &);
...
```

2.- Los operadores se dividen en dos grandes grupos: operadores unarios (un sólo operando) y binarios (dos operandos).

3.- La definición de operadores unarios suele hacerse a través de métodos miembros de la clase; es decir, no friend.

4.- Cuando sobrecarge un operador binario tenga en cuenta que puede llegarse a ejecutar al revés. Es decir, el operador "+" debe servir tanto para sumar un complejo con un float como un float con un complejo.

5.- Si las dos clases que intervienen en un operador binario son creadas por el usuario, sobrecargue el operador en la primera para que reciba un objeto de la segunda, y en la segunda para que haga lo propio con uno de la primera.

6.- Si una clase es propia y la otra no, sobrecarge el operador dentro de la propia dos veces; una como mé-

```
...
Complejo &Complejo::operator = (const
Complejo &) {
        ...
        return (*this);
}
Complejo &Complejo::operator += (const
Complejo &) {
        ...
        return (*this);
}
...

void main (void) {
        ...
        Complejo A,B,C;
        C = A = B;
        C = A += B;
        ...
}
```

todo miembro (para que reciba un objeto de la otra) y el otro como método friend (para que reciba, primero, un objeto de la no propia y, luego, uno de la suya). Se asegura así, la reversibilidad del operador.

7.- Todos los parámetros recibidos por los métodos de sobrecarga de operadores deben ser const &, salvo si dicho parámetro pertenece a un tipo estándar (float, int, etc...). Hacer esto aumenta la velocidad (no hay que hacer copias en la pila de los objetos parámetros del método).

8.- Todos los métodos de sobrecarga de operadores, a excepción del "=" y los del estilo del "*=", han de devolver un nuevo objeto. Los excluidos, por el contrario, devolverán una referencia, no constante, al objeto que los invocó. Tanto uno como el otro deben hacerlo para poder permitir la encadenación de operaciones tan típica del C.

9.- Si sobrecarga un operador, haga lo propio con todos aquellos que son semejantes, si es que tiene, por supuesto, sentido hacerlo. Es decir, si sobrecarga el operador mayor para comparar dos objetos cualesquiera, haga lo propio con "==", "!=", "<", ">=", "<=", etc... Aunque ahora no los vaya a emplear, puede hacerlo bien más adelante, bien en otra aplicación. La reusabilidad de su clase aumenta.

10.- Sobrecarge un operador sólo en el caso de que tenga sentido lógico hacerlo. Es decir, no emplee, por ejemplo, el operador "==" para restar dos objetos. Iría en contra del objetivo pretendido de facilitar la legibilidad y el mantenimiento.

11.- Sobrecarge siempre tanto el operador de inserción ("<<") como el de extracción (">>") para permitir la entrada/salida de sus objetos. Sólo puede hacerse mediante funciones friend.

Los constructores y destructores han de ser públicos

12.- La sobrecarga de operadores se hereda. Es decir si dentro de una clase sobrecarga el operador "+" para sumar objetos de la misma y lo emplea para sumar objetos de una clase hija, se usará ese operador definido en la padre, siempre y cuando, claro está, no la haya redefinido nuevamente en la hija.

13.- Los métodos de sobrecarga de operadores no admiten la virtualidad.

HERENCIA

1.- Para declarar una clase hija de otra, coloque tras la declaración de la clase (class) y seguido de dos puntos el nombre de la clase padre precedido de la forma en la que se realiza esa herencia (public, protected o private).

Las plantillas template pueden ser incluidas en librerías

2.- Es difícil encontrar una situación en la que convenga realizar un tipo de herencia no pública, aunque, claro está, lo tiene a su disposición.

3.- No utilice la herencia como una forma de compartir atributos o métodos. Hágalo sólo cuando sea lógico desde el punto de vista conceptual. Dicho punto de vista dependerá de la aplicación en si.

4.- Si una clase es hija de otra, para definirla tiene que estarlo antes la padre (include).

5.- Procure no alcanzar nunca profundidades en la herencia superiores a los 4 ó 5 niveles. Cualquier otra persona distinta al desarrollador encontrará dificultades para comprender un código semejante (límite de Miller).

6.- Aunque el C++ lo permite e, incluso, lo usa en sus propias librerías, procure no emplear la herencia múltiple (dos o más padres) ya que suele presentar muchas situaciones de ambigüedad y dificulta, además, la lectura del código. La mayor parte de los productos para el control de la calidad del software también lo desaconsejan.

7.- No altere nunca, a lo largo de una herencia, el ámbito de definición de los métodos virtuales. Es decir, si en la clase raíz un método virtual fue definido como public, debe conservar ese ámbito a lo largo de todas sus redefiniciones en las clases hijas

```
class ClaseA {
        public :
                ...
                virtual tipo METEODOA (parámetros) = 0;
                ...
};

class ClaseB : public A {
        public :
                ...
                virtual tipo METODOA (parámetros);
                ...
};
```

VIRTUALIDAD

1.- Nunca se pueden crear instancias de clases que contengan algún método virtual puro. Esas clases se denominan abstractas.

2.- Si en una clase hija, no se definen los cuerpos de los método virtuales puros, dicha clase sigue siendo abstracta y, por tanto, tampoco se pueden declarar instancias de ella.

3.- Aunque desee que una clase hija de otra abstracta sea también abstracta, redeclare los métodos vir-

tuales puros que no vaya a redefinir. Es una manera de no olvidarse nunca de ninguno en las clases más bajas de la herencia.

4.- Cuando un método virtual se redefine en una clase hija, no es necesario precederlo nuevamente de la palabra clave virtual. A pesar de ello, hágalo ya que facilita la legibilidad de sus clases.

5.- Los métodos static, friend y constructores no pueden ser declarados como virtual.

TEMPLATES

1.- Sólo cuando se declara alguna instancia de la clase template, puede asegurarse que dicho template podría ser usado como plantilla. Hasta ese momento el compilador únicamente verificará su sintaxis.

2.- Se pueden construir librerías sólo con definiciones template. No es necesario, por tanto, declarar instancias para hacerlo.

3.- Emplee siempre un fichero .cpp para definir todas las instancias template empleadas por su aplicación. De esta forma podrá emplear la misma plantilla entre distintas aplicaciones y definir en ellas las instancias que más convengan.

4.- Las instancias de clases template no se crean igual en todos lo compiladores. Esto ratifica la idea de declarar las instancias en un fichero distinto.

CONSEJO FINAL

Las normas y consejos que aquí se han expuesto son sólo los más relevantes. Existen otros muchos cuya exposición ocuparía bastantes más hojas. La mayor parte de ellos son razonables y lógicos, y pueden ser descubiertos por usted a través de la experiencia. No tenga estas normas como obligatorias, sino más bien como consejos de los más iniciados en la materia. Un saludo. ■

Emplee la sobrecarga de operadores con moderación

BIBLIOGRAFÍA

Tkach-Puttick: Object Technology in Application Development. Ed. IBM INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION.

Lippman, Stanley B.: C++ Primer. Ed. ADDISON-WESLEY, 2ª Edición.

Para programadores en CA-Clipper

Product líder Five

No lo dudes. Este producto es el que necesitas para hacer tu aplicación en Windows. Extremadamente fácil y potente. Resultados inmediatos. Más de 4.000 usuarios en todo el mundo.

FiveWin
Librería de CA-Clipper para Microsoft Windows
Realiza tus aplicaciones en Microsoft Windows con una presentación y funcionalidad totalmente profesionales. Elegido por la revista Americana Reference(Clipper) como el mejor producto para xBase en Windows (enero 94)
Precio promocional: 14.000 pts.

FiveOS2
Librería de CA-Clipper para IBM-OS2
Ahora puedes realizar aplicaciones de gestión para IBM-OS2 con la misma facilidad que para Windows. El entorno IBM-OS2 es el máximo competidor de Windows y se está haciendo muy popular. Aprovecha esta oportunidad.
Precio promocional: 14.000 pts.

FiveDos
Librería de CA-Clipper para MsDos
Un completo entorno de desarrollo para MsDos con ventanas, ratón, controles CUA y ejecución NoModal. Simula totalmente la funcionalidad de Windows y OS2.
Precio promocional: 14.000 pts.

¡ FiveWin en Vídeo !
Aprende ahora rápida y cómodamente la manera de realizar aplicaciones profesionales de gestión en Windows, tan fácilmente como ver una película de vídeo
Volumen 1.....8.000 pts.

Antonio Linares - Software
Urb. El Rosario, Avda. Rosario 34-A. 29600 Marbella - España
Tfno./fax: 95-2834830 BBS: 95-2213374 / 908 453368 voz

Distribuidor Nacional
Mail Simons S.L. Apdo. 2643. 28080 Madrid. Tfno. 91-5634486
BBS: 91-5637872 FAX: 91-5634451 E-mail: bmartinez @ simons.es

FiveWin

CA-Clipper es una marca registrada de Computer Associates, Windows es una marca registrada de Microsoft Corporation, OS2 es una marca registrada de IBM

DBU para Windows: Gestor de ficheros dBase para programadores. Soporta índices Clipper NTX, dBase III, dBase IV, Foxpro (Comix, Six 2.0) y NSX. PVP: 8.500 Ptas. FORMACIÓN, ASESORAMIENTO y CURSOS de programación WINDOWS con FiveWin en Madrid: ORTIZ DE ZÚÑIGA. Tfno: 91-575 20 47.

OTROS DISTRIBUIDORES

ARGENTINA
GO CLIPPER BBS&
FIVEWIN
Panamericana 3787
1609 Boulogne
ARGENTINA
+541-737-2767 Voz
+54-1-943-0563 Fax
+54-1-790-1906 BBS

BRASIL
VENEZUELA
CUBA
PARAGUAY
URUGUAY
SE BUSCA
DISTRIBUIDOR

CHILE
PROGRAMMER'S
HOUSE
Bustos 2157 dpto 13
Providencia/
Santiago
CHILE
+56-2-2322948 Voz
+56-2-2322948 Fax

CHILE
COMPUDATA
TUCAPEL 564 of. 77
CONCEPCIÓN
+56-41246035 Voz
+56-41542184 Fax

MÉXICO
PERU
COLOMBIA
BOLIVIA
SE BUSCA
DISTRIBUIDOR

PORTUGAL
MULTDIGITAL LDA
Pr. Manuel Guedes,
13 - 4º - Sala 17
+351-2-4647050 Voz
+351-2-4647051 Fax
+351-2-4647052 bbs
joaquim boavida @ tail. pt

La última versión la puedes encontrar en las siguientes revistas, CD-ROM's, BBS's o direcciones de INTERNERT: CD-ROM de PC MEDICA, Sólo Programadores, PC-ACTUAL, KENDER, AMS (Mail Simons s.l.) y las siguientes BBS's: bAuHaUs (+34-5-2213374), KENDER (944763506), CIBERLINEA (902-238624)., CIBERNETIX (986-691525) ftp. eunet. es: pub/InterStand/Simons.

SLACKWARE LINUX 2.2.0.1

Con este ejemplar de Sólo Programadores se adjunta un CD-ROM con una interesante selección de programas y utilidades. Cada uno de ellos se encuentra situado en un directorio distinto para facilitar su acceso al lector.

En primer lugar se ha incluido una nueva versión del cada vez más popular sistema operativo Linux. Conjuntamente se incluye una distribución reducida, denominada Mini Linux, que permite al usuario una sencilla instalación y un acercamiento a este sistema operativo sin necesidad de crear nuevas particiones en el disco duro. También se incluye una recopilación de programas de utilidad para Linux.

Por otra parte, y fuera del ámbito del sistema operativo Linux, se suministran otros interesantes programas y herramientas para DOS y Windows, como por ejemplo la última versión de las librerías FIVEWIN.para Windows, DOS y OS/2.

LINUX 2.2.0.1

En el directorio *LINUX220* puede encontrarse una nueva versión del sistema operativo Linux. Se trata de la distribución Slackware Linux 2.2.0.1, que supone una importante mejora de la mayoría de los aspectos de este sistema operativo.

Esta versión contiene el Kernel 1.2.3 de Linux (incluyendo el código fuente de la mayoría de las versiones anteriores, y de la versión 0.01). También se incluyen las librerías Libc 4.6.27 y una nueva versión del entorno X-Window, XFree86 3.1.1.

Para realizar la instalación de esta nueva versión, el lector puede consultar el libro que acompaña al número 9 de la revista, si bien, este proceso es virtualemente idéntico al de versiones anteriores.

MINI LINUX

En el directorio *MINILIN* se suministra una distribución especial del sistema operativo Linux. Se trata de un sistema reducido de distribución (sólo 4 disquetes) que permite una fácil instalación, especialmente para los usuarios de MSDOS. A pesar de ello, cuenta con todas las herramientas y la potencia de Linux (TCP/IP y SLIP, X-Window, etc).

La instalación estándar de Linux obliga al usuario a realizar un Backup del disco duro, así como una repartición y un formateado del mismo. Sin embargo, esta distribución de Linux permite instalar el sistema operativo en una partición del MSDOS, coexistiendo simultáneamente con éste. Para ello, en la parte del DOS se crea un directorio *C:\LINUX* que contiene el comando "linux", y que permite acceder a Linux de forma automática. Igualmente, en la parte de Linux, aparece el directorio */DOS* que contendrá todos los ficheros del DOS. Para volver al DOS será necesario reinicializar el ordenador (pulsando CTRL+ALT+DEL).

Además, esta distribución ofrece un sistema basado en Unix con acceso a Internet. Para ello, se suministra todo lo necesario para crea un enlace SLIP con el servidor. También se incluye el programa Mosaic, el más popular de los programas WWW para la navegación por Internet.

INSTALACIÓN DE MINI LINUX

Para la instalación de Mini Linux es necesario disponer de 20Mb de espacio libre en la unidad C:, si bien éste sólo ocupa 15Mb una vez finalizado el proceso.

La distribución se compone de 4 ficheros ZIP (de MINILIN1.ZIP a MINILIN4.ZIP) y el fichero MINILINX.EXE. Para su instalación debe crearse el subdirectorio LINUX en el directorio raiz de la unidad C: y copiar los ficheros anteriores en el mismo. Posteriormente debe ejecutarse el programa MINILINX.EXE para obtener las utilidades encargadas de descomprimir los ficheros ZIP y completar la instalación.

```
C:\>md linux
C:\>cd linux
C:\LINUX>minilinx
```

Una vez obtenidas dichas utilidades, deben descomprimirse los archivos ZIP, tecleando lo siguiente en la línea de comandos:

```
C:\LINUX>pkunzip -d *
```

En este momento es posible eliminar los ficheros ZIP del directorio, con el fin de ahorrar espacio. Una vez finalizada esta operación, se habrán creado el fichero LINUX.CUT y los ficheros LINUX.001 a LINUX.003.

CD-ROM

Estos ficheros deben unirse para formar el fichero final de la instalación, LINUX.ZIP, mediante la utilización del programa FCUT.

```
C:\LINUX>fcut -j linux
```

Una vez unidos, es posible eliminarlos, para ahorrar espacio. La instalación concluye con la descompresión del fichero LINUX.ZIP.

```
C:\LINUX>pkunzip -d linux
```

Para acceder a Linux bastará con ejecutar el fichero LINUX.BAT, que desencadena el arranque de este sistema operativo. El primer acceso debe realizarse como superusuario, introduciendo *root* en el login sin *password*.

UTILIDADES LINUX

Como complemento al sistema operativo Linux se ha incluido un grupo de programas de utilidad. Estos programas, que se encuentran disponibles en el directorio VARIOS, son los descritos a continuación:

dosemu.tgz. DOSEMU 0.60.1 Emulador de DOS para Linux.

os2win31.zip. Permite ejecutar las Windows desde DOSEMU. Estos módulos fueron desarrollados por IBM para el OS/2, aunque también funcionan en Linux.

lnx127.tgz. Versión 1.2.7 del kernel de Linux.

acm.tgz. Simulador de vuelo para **X-Windows**. En código fuente.

flysrc.tgz. Otro simulador de vuelo para X-Windows. Este se ha diseñado siguiendo una filosofía diferente de la utilizada en el ACM.

tagl21sr.tgz. Librería de gráficos en 3D. Al igual que las VOGL, son un clónico que las Open-GL, aunque con mas prestaciones que las ya mencionadas VOGL.

mpeg.tgz. Librerías para manejo de ficheros MPEG.

joyfixed.tgz. Última versión del driver de JoyStick para Linux.

fs.tgz. Versión demo del FlagShip. El FlagShip es un compilador de Clipper para Linux.

samba.tgz. Soporte SMB para Linux. Permite utilizar el Linux como servidor en una red con maquinas basadas en Windows 3.11.

hdparm.tgz. Utilidad para fijar parámetros de la controladora IDE. Para mejorar el rendimiento de las unidades de disco.

gnupas.tgz. Compilador de Pascal de GNU. Distribución en binario.

lsm.gz. Linux Software Map. Lista de programas para Linux disponibles en Internet. Es un fichero de texto.

lsmtools.tgz. Utilidades para examinar cómodamente el fichero LSM.

WINDOWBASE

En el directorio WINBASE se incluye el programa WindowBase 2.0 de SPI. Se trata de un gestor de bases de datos para el entorno Windows de muy fácil manejo.

Update now!

WATCOM

Compiladores para lenguajes profesionales
- Lenguaje C/C++V.10.0 CD ROM
- Lenguaje Fortran 77/32 Bits v 9.5

Ambiente de desarrollo visual (Front End)
- Lenguaje OS/2 Rexx
- VX-REXX v.2.1 Standard Edition
- VX-REXX v.2.1 Client/Server

Bases de datos relacionales (llamada SQL) v 4.0
- Plataformas: Windows, NT, OS/2, DOS y Netware
- Usuarios: Standalone - Network Server
- Lenguaje: C/C++

RAIMA

Raima Database Manager: Base de Datos
- Lenguaje: C
- Sistema Operativo: DOS, Windows, OS/2, UNIX System V, Berkeley 4.2 AIX, Sun OS, SCO, QNX, YULTRIX y VMS
- Plataformas: DOS, OS/2
- Modalidades: Module y System
- Royalty Free

VELOCIS

Velocis: Base de datos. Tecnología Cliente/Servidor
- Lenguaje C/C++
- Sistema Operativo: Windows, NT y UNIX
- Plataformas: Windows, NT, OS/2, Netware 386 NLM, OS/2 y UNIX
- Velocidad de acceso 1/3 más rápido que otras B/Datos

PHARLAP

Gestiona la memoria optimizando los límites de todas las versiones DOS

SEQUITER Software Inc.

Librerías de programas para trasladar aplicaciones de Dbase a C/C++
- CodeServer v.5.1 Cliente/Servidor

OFERTA ESPECIAL

Para los lectores de SÓLO PROGRAMADORES

Infórmate llamando:
Tel.: 485 13 04
Fax: 485 14 29

C. Lope de Vega, 21 bjs.
08005 Barcelona

Permite realizar las tareas de diseño y manipulación de las bases de datos de una forma visual. Además, las consultas SQL y QBE pueden realizarse de una forma sencilla e intuitiva. Este sistema permite una gran conectividad, que resulta prácticamente transparente al usuario.

Para instalar el programa debe disponerse al menos de 5Mb de espacio en el disco duro. Para ello bastará con ejecutar desde el administrador de programas de Windows el fichero SETUP.EXE del directorio DISK1, ya que el proceso de instalación está totalmente automatizado.

PROGRAMAS DE GESTIÓN

El directorio *MSIMONS* incluye los subdirectorios *INMOBIL*, *ABOGADOS* y *JURIDICO*. El primero almacena un programa para la gestión de inmobiliarias, el segundo contiene software que permite organizar el funcionamiento de un despacho de abogados y, por último, en el tercero se puede encontrar un programa de recuperación de textos jurídicos.

LIBRERÍAS CLIPPER PARA WINDOWS

En el directorio *FIVETECH* se encuentran cinco subdirectorios con herramientas para facilitar la programación en CA-CLIPPER en los entornos Windows, DOS y OS/2. Asimismo se incluye una gran cantidad de ejemplos, que ilustran el manejo y posibilidades de dichas herramientas.

Dentro de *FIVEWIN* se halla la versión 1.8E de las librerías del mismo nombre. Estas facilitan en gran manera la generación de aplicaciones Windows. De la misma forma, *FIVEDOS* y *FIVEOS2* contienen las correspondientes librerías para DOS y OS/2. En el directorio *WINDBU* tenemos un potente gestor de bases de datos con soporte para distintos tipos de índices. Y, por último, en el subdirectorio *FIVEEJEM* se encontrarán ejemplos de todo lo anterior.

COMUNICACIONES

En el directorio *TOWERTEX* se ha incluido un programa de comunicaciones que permite el acceso a la red IBERTEX. Se trata del programa VtxCtr, un decodificador de Videotex (norma CEPT1) compatible con la red española IBERTEX. Con él es posible acceder a los distintos centros servidores (especialmente al de TOWER Communications) y beneficiarse de las múltiples utilidades que estos proporcionan. Con este programa es posible realizar transferencias de ficheros, Teleimpresión automática, etc. Soporta varios tipos de módem y gráficos en alta resolución.

También se ha incluido otro programa de comunicaciones en el directorio *TERMINAT*. Se trata del programa Terminate 1.58h. Este programa añade varias utilidades a las funciones estándar de los programas de comunicación. Su manejo es sencillo y puede realizarse con el ratón. Además, puede funcionar en modo terminal o FAX.

SHAREWARE

En el directorio *SHARE* se encuentra el programa HyperNoticias 95 de AACHE Ediciones, que ilustra el funcionamiento y las posibilidades del Hypertexto como vehículo de comunicación y como soporte para la información. Este programa puede ser ejecutado directamente desde el CD-ROM sin necesidad de instalación previa, ejecutando para ello el fichero AACHE.EXE.

Se debe reseñar que la editorial AACHE organiza un concurso sobre la provincia de Guadalajara, sorteando 25 libros entre los acertantes de tres preguntas, contenidas en el archivo CONCURSO.TXT.

Este concurso carece por completo de finalidad comercial, y responde tan sólo al deseo de promocionar la provincia de Guadalajara. Los participantes tienen garantizado en todo momento que sus datos no serán incluidos en ningún sistema de proceso de datos.

RESULTADOS DEL TORNEO

Como se prometió en el número anterior, en el CD-ROM se incluyen los resultados de las partidas de "4 EN RAYA" que tuvieron lugar entre los programas enviados por nuestros lectores.

Dentro del directorio *TORNEO* se encuentran los subdirectorios *LOGS1, LOGS2,...,LOGSF*, y dentro de ellos los archivos de texto que muestran qué enfrentamientos tuvieron lugar y el resultado de cada uno. Cada participante está identificado mediante el número con el que apareció en el listado de jugadores del nº 9 de la revista.

Por otro lado en el directorio *GANADOR* se encuentran los fuentes y ejecutables de los cinco primeros clasificados. Los fuentes se presentan comprimidos en formato .ZIP, y el nombre de los archivos hace referencia al puesto definitivo en el que quedaron. El fuente y el ejecutable del programa que se empleó para realizar los combates se encuentran dentro del directorio *ARBITRO*. Este programa, ARBITRO.EXE, como se dejó expuesto en el nº 9 de "Sólo Programadores", genera los archivos *.LOG en los que se reflejan los resultados de las partidas jugadas.

Lamentablemente, en los desarrollos de las partidas aparecidos en el número anterior, se deslizaron una serie de errores en el momento de transcribir los resultados de cada encuentro, y esto, evidentemente, produjo una comprensible confusión en algunos participantes, a los que pedimos disculpas. El contenido de los archivos *.LOG muestra que todos los programas enviados participaron en el torneo de programación.

REVISTA

Por último, también se han incluido en el CD-ROM los programas que acompañan a los artículos de este número. En el directorio *REVISTA* se encuentran los subdirectorios *APDMX, ASM, CLIPPER, CURSOP, FORMATOS, LINUXVGA, NETBIOS* y *XMS*, que almacenan los archivos fuente, objeto y/o ejecutables de los ejemplos que acompañan a cada uno de esos artículos. ❑

CORREO DEL LECTOR

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relaccionado con la programación.

Tengo problemas con la paleta del modo 640x480 de 16 colores, ya que los colores "por defecto" son insuficientes y poco apropiados, también quiero poner una pantalla PCX en una zona de la ventana, pero me veo perdido en el proceso. Si pudieran ayudarme les estaría muy agradecidos.

Agustín Ortega Cintas
(Madrid)

Por lo que cuenta en su carta (más extensa que la pregunta anterior), le aconsejaría que utilizara un modo de 256 colores, pues hoy en día este modo es estándar mediante la norma VESA, si su tarjeta no viene con dicha norma incluida (casi todas las tarjetas modernas la traen), puede cargarla externamente mediante un pequeño programa muy extendido (en casi todos los CD distribuidos por Tower) llamado UNIVESA.EXE. Los modos de las resoluciones parecidas son:

Valor	Modo gráfico
100h	640x400x256 colores
101h	640x480x256 colores

Sobre la lectura y manejo de ficheros PCX, en un número anterior de la revista viene explicado su formato y un programa de ejemplo que se encarga de leerlos y mostrarlos. Fácilmente puede modificarse para que los muestre en otra resolución y en una determinada posición de la pantalla.

Les escribo porque tengo un problema con los ejemplos incluidos en la serie de artículos sobre programación en 3D que están publicando en su/nuestra revista. Al ejecutar los últimos ejemplos, aparecen unas líneas verticales en la pantalla que impiden ver cualquier cosa, además se superponen los planos de manera descontrolada. ¿A que se puede deber?

Jesús Díaz Carballo
(La Coruña)

Se trata de un problema de incompatibilidad entre las rutinas de modo gráfico X y su tarjeta de vídeo. Estas rutinas realizan acceso al hardware de vídeo a muy bajo nivel y al parecer surgen problemas con algún fabricante. En este número y en los siguientes se han retocado dichas rutinas para intentar solventar el problema, pero si persiste habrá que utilizar modos gráficos estándar aunque la velocidad se vea afectada.

Necesito información sobre como programar y manejar scanners, pero hasta el momento no he encontrado apenas nada. ¿Existen rutinas más o menos estándares sobre ello, de manera parecida al manejo del ratón?.

Manuel E. Cotallo Torres
(Cáceres)

Los distintos fabricantes de scanners nunca se pusieron de acuerdo sobre una posible estandarización de sus productos (o al menos sobre la manera de programarlos). Pero desde hace algún tiempo, prácticamente todos vienen con un driver a bajo nivel llamado TWAIN, que les permite un acceso idéntico por parte de las aplicaciones Windows que acceden a ellos. Dichas aplicaciones solo tienen que comprobar y manejar el scanner mediante dicha norma incluida en Windows (TWAIN.DLL), para así poder manejar cualquier scanner o dispositivo similar conectado al ordenador. Si el scanner a controlar no dispone de dicha norma, el único modo de manejarlo es pidiendo al propio fabricante la información técnica referente a su producto, usualmente no suelen poner obstáculos a enviarla.

¿Que diferencias existen entre una plataforma cliente/servidor y una maestro/esclavo? ¿Porqué no publican más artículos sobre Basic?, yo habitualmente trabajo sobre QuickBasic 4.5 de Microsoft y sé de muchos otros que también desarrollan bajo alguna de las versiones de tan popular lenguaje (GW-Basic, Qbasic, TurboBasic, Basic PDS).

Javier Polo Sánchez
(Cáceres)

Sobre lo del Basic, supongo que tiene razón, tomaremos nota y a ver si en futuros números incluimos algún artículo o documentación; aunque hay que tener en cuenta que en la actualidad el Basic se está relegando un poco (por lo menos el Basic clásico, por algunas versiones en el mercado son "superlenguajes" que poco o nada tienen que ver con los orígenes), al expandirse el C. Me parece que las designaciones informáticas son más bien pura filosofía e intentar definir que significa tal o cual cosa se convertiría en una tarea prácticamente imposible, pero según mi modesta opinión un sistema maestro/esclavo consiste en simples terminales conectados a un host que se encarga de "darles vida" por completo, mientras que una plataforma

cliente/servidor se refiere a aplicaciones o sistemas (clientes) que acceden a un sistema central (servidor) para recolectar datos o diversos servicios. Un claro ejemplo de la filosofía cliente/servidor sería el de las redes centralizadas alrededor de uno o varios servidores, siendo los nodos de los usuarios los clientes.

Estoy intentando programar mi tarjeta de sonido compatible SoundBlaster 2.0, pero al inicializar el DSP no obtengo el valor deseado AAh ¿A que se puede deber ? Gracias de antemano por su respuesta.

A. Pulido Alario
(Madrid)

El procedimiento para reinicializar el DSP es el siguiente:

1.- Escribir el valor 1 en el puerto RESET (2x6h) y esperar 3 microsegundos.
2.- Escribir el valor 0 en el mismo puerto.
3.- Obtener el valor AAh (Ready Byte) en el puerto READ DATA (2xAh). Pero antes de leer este puerto, es aconsejable el chequear el bit DATA AVAILABLE en el puerto 2xEh (bit más significante). Si después de 100 bucles, no se obtiene el valor AAh (o ninguno), es que ha habido algún problema.

El DSP suele tardar unos 100 microsegundos en inicializarse.

Si el problema persiste compruebe que está utilizando la dirección correcta para los puertos de la tarjeta.

Los Protectores de Pantalla tan habituales en Windows, ¿de qué manera benefician al monitor?.

Santiago Varcarcel
(Toledo)

La función de los Protectores de Pantalla es evitar un desgaste prematuro del fósforo que recubre la parte interior de la pantalla del monitor en las zonas que permanecen más claras (iluminadas) durante mucho tiempo. Lo realizan cambiando constantemente el aspecto de la pantalla para evitar que ciertas zonas se desgasten más que otras (es el caso habitual en las pantallas de los cajeros automáticos). En la actualidad, con la tecnología Green Card que incorporan casi todas las placas base actuales, consistente en desconectar la señal de vídeo y literalmente "apagar" el monitor, los días de los "salva pantallas" están contados, a no ser por motivos estéticos, que también cuentan...

¿Cual es la técnica para superponer un gráfico sobre otro sin estropear el inferior? Supongo que es la base para hacer cualquier tipo de animación por ordenador.

José Tejada Gómez
(Córdoba)

La teoría es muy sencilla, pero hacerlo bien (¡y rápido!) en la práctica no lo es tanto. Básicamente consiste en los siguiente pasos:

- Dibujar el fondo sobre el cual van a ir montados los gráficos o sprites.
- Dibujar uno tras otro los sprites, empezando por el que aparente estar más lejos, esto es, el que vaya a pasar por detrás de los otros. Si hay multitud de gráficos que se mueven incluyendo en eje Z (profundidad con respecto a la pantalla), es necesario intercalar una rutina que los ordene del más lejano al más cercano, antes de empezar a dibujarlos. La forma de "pintar" los sprites es con transparencias, pues debe poder verse el fondo y a los otros sprites a través de las partes sin color. Si esto no se hiciera así tendríamos un montón de feos rectángulos moviéndose y pisándose por la pantalla. Para conseguir las transparencias, se debe asignar un color (usualmente el número 0 y en negro) como "tinta transparente" al crear el gráfico y a la hora de pintarlo en pantalla, basta con saltarnos los pixels de este color.

Todo esto se puede hacer directamente en la pantalla real o física (dirección A0000h), pero no es aconsejable, pues se verían todos los procesos de creación de la pantalla. Habitualmente se realiza en una zona de memoria auxiliar y una vez terminado el montaje, se vuelca a la pantalla física.

Claramente el dibujar cada vez el fondo completo y luego todos los sprites es lento, por lo que se utilizan multitud de trucos para acelerar el proceso, tantos como programadores creando animaciones.

Sigo con impaciencia el curso de programación en 3D, pero desde que se empezó a utilizar el modo gráfico X (creo que para ganar velocidad y para evitar parpadeos), no me aparecen correctamente los gráficos. ¿A que se puede deber? Mi tarjeta gráfica es una OAK por si les es de interés.

Isidro Villarroya Fernández
(Torrente - Valencia)

Efectivamente, los modos X se utilizan para ganar velocidad, para permitir un mejor intercambio entre pantallas y para poder tener resoluciones mayores en pantalla, como 320 x 240 pixels. Estos modos no están documentados oficialmente pero prácticamente todas las tarjetas de vídeo los soportan...menos un tipo fabricado por OAK Technology, Inc. Para evitar los problemas que puedan tener los usuarios de estas tarjetas, se están recodificando todas las rutinas gráficas del curso de 3D para volver a utilizar el modo estándar 13H (320 x 200). Con él nos aseguramos una total compatibilidad, aún sacrificando prestaciones.

TECNO CD

CD ROM

Líder europeo en fabricación de soportes ópticos

En Tecno CD la perfección tecnológica, unida al servicio más flexible, pone al alcance tanto de los grandes, como de los más pequeños, la máxima calidad en CD ROM.
Nuestra gran capacidad de producción diaria, con la máxima rapidez, a partir de los pedidos más reducidos, abre el círculo.
Tecno CD, como filial en España de MPO, líder internacional en la tecnología y producción de soportes ópticos, agrupa la totalidad de los soportes CD, CD Audio, Laser Disc, CD Video, CD Interactivo y CD ROM.
Hoy Tecno CD, después de tan sólo tres años de actividad, cuenta con el 60% de la fabricación de CD Audio en España.(Mastering y 5 líneas de fabricación).
Es el reflejo de nuestro espíritu de perfección en Tecnología y Servicio.

TECNO CD. Anticipamos el Futuro

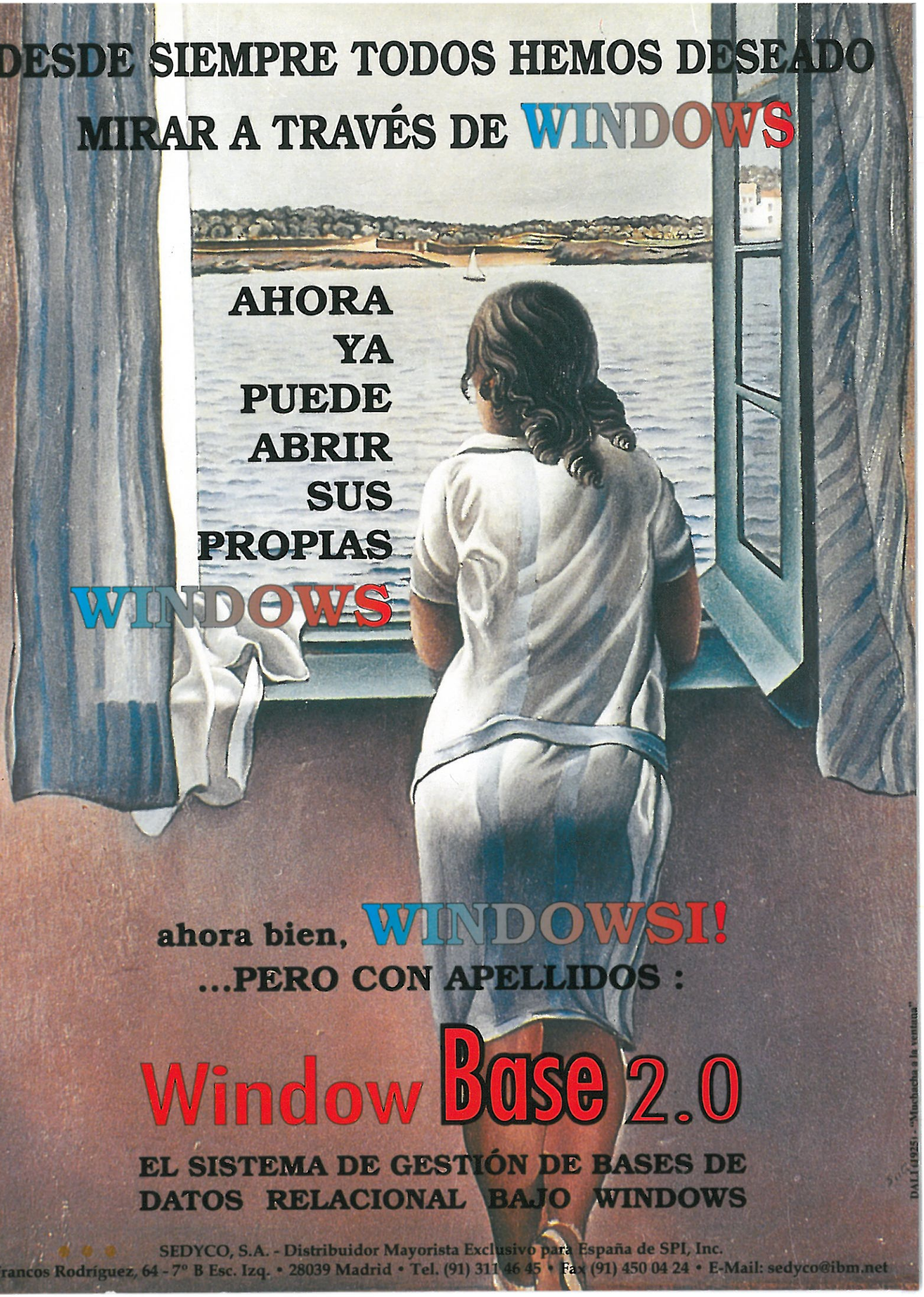
MPO

MASTERING Y DUPLICACION
TECNO CD

La Laguna, 33. Políg. Ind. Urtinsa
28925 ALCORCON (Madrid) España
Tel.: (91) 643 12 38 - Fax (91) 643 02 38
Gran Via de les Corts Catalanes, 617.1°, 3ª
08007 BARCELONA
Tel.: (93) 412 29 45 - Fax: (93) 412 44 41

CD ROM CD AUDIO LASER DISC CD VIDEO CD INTERACTIVO MINI DISC

DESDE SIEMPRE TODOS HEMOS DESEADO MIRAR A TRAVÉS DE WINDOWS



AHORA YA PUEDE ABRIR SUS PROPIAS WINDOWS

ahora bien, WINDOWSI!

...PERO CON APELLIDOS :

Window Base 2.0

EL SISTEMA DE GESTIÓN DE BASES DE DATOS RELACIONAL BAJO WINDOWS

© SEDYCO, S.A.

DALÍ (1925) - "Muchacha a la ventana"

SEDYCO, S.A. - Distribuidor Mayorista Exclusivo para España de SPI, Inc.
Francos Rodríguez, 64 - 7° B Esc. Izq. • 28039 Madrid • Tel. (91) 311 46 45 • Fax (91) 450 04 24 • E-Mail: sedyco@ibm.net